Iz oblasti elektronike raspolažemo i sledećim izdanjima:

N. Mladenović, R. Grbović, V. Petrović
KUĆNI KOMPJUTERI — ALGORITMI I PROGRAMI
Format 17 × 24 cm. Str. 192

B. Đurić
MINI I MIKRO RAČUNARI
Format 14 × 20 cm. Str. 472

V. Cvekić
INTEGRISANA KOLA
Format 14 × 20 cm. Str. 288

D. Pantić, J. Pešić
PRIMENA LINEARNIH INTEGRISANIH KOLA
Format 14 × 20 cm. Str. 412

D. Pantić, J. Pešić
PRIMENA DIGITALNIH INTEGRISANIH KOLA
Format 14 × 20 cm. Str. 276

V. Cvekić
POLUPROVODNIČKE DIODE I TRANZISTORI
Format 14 × 20 cm. Str. 388

B. Đurić
TIRISTORI
Format 14 × 20 cm. Str. 426

Grupa autora
OSNOVI PROJEKTOVANJA INFORMACIONIH SISTEMA ZASNOVANIH NA PRIMENI RAČUNARA
Format 14 × 20 cm. Str. 324

M. Cvekić, D. Basić
MIKROGRAFSKI SISTEMI
Format 14 × 20 cm. Str. 308

Grupa autora
ELEKTRONSKI MERNI INSTRUMENTI
Format 14 × 20 cm. Str. 308

Knjige se mogu nabaviti u svim knjižarama ili kod izdavača:

NIRO »TEHNIČKA KNJIGA«
Beograd, 7. juli 26

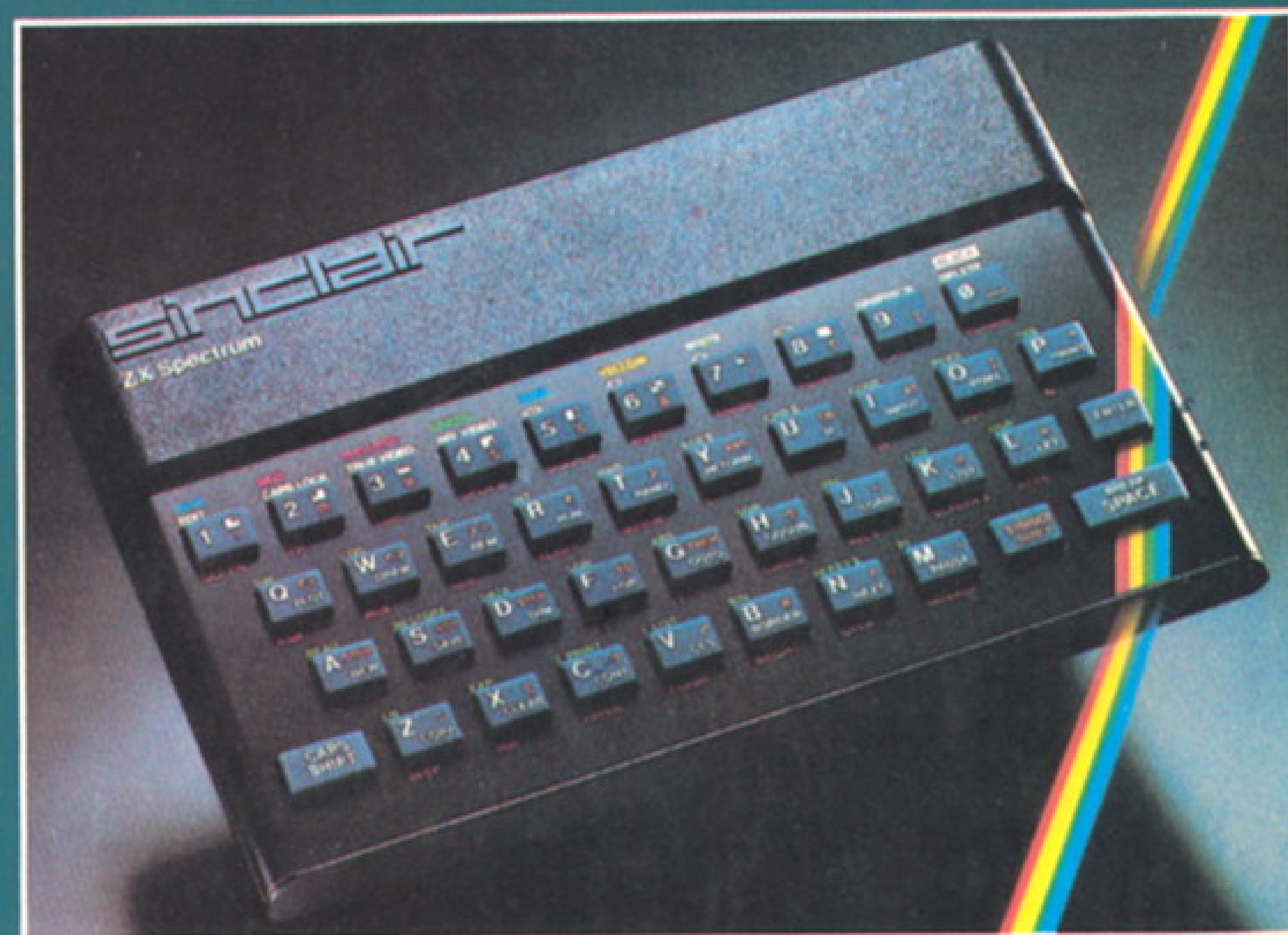
ZX SPECTRUM

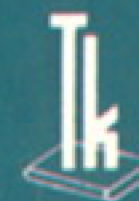
N. Marković - D. Davidovac

N. Marković - D. Davidovac

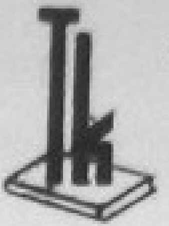
ZX SPECTRUM

Programiranje u BAZIC-u





Mr NENAD MARKOVIĆ, dipl. inž.
DUŠAN DAVIDOVAC, inž. inf.



ZX SPECTRUM

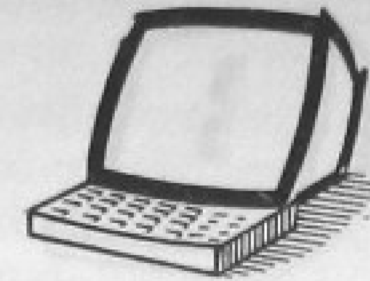
programiranje u BASIC-u

Mr NENAD MARKOVIĆ, dipl. inž.
DUŠAN DAVIDOVAC, inž. inf.

ZX SPECTRUM

Programiranje u BASIC-u

NIRO „TEHNIČKA KNJIGA“ I ZAVOD ZA IZDAVANJE UDŽBENIKA
— OOUR STVARANJE I PROIZVODNJA NASTAVNIH SREDSTAVA —
BEOGRAD, 1985.



Predgovor

Knjiga ZX SPECTRUM — Programiranje u BASIC-u namenjena je svima onima koji poseduju lični računar ZX SPECTRUM, nameravaju da ga nabave ili poseduju neki drugi lični računar koji koristi programski jezik BASIC. Kako je krug potencijalnih čitalaca veoma širok, a posebno izuzetno raznolikog znanja iz programiranja, namera nam je bila da zadovoljimo što veći broj interesenata za programiranje i korišćenje računara ZX SPECTRUM. Zato je u knjizi dovoljno prostora posvećeno i postupcima programiranja.

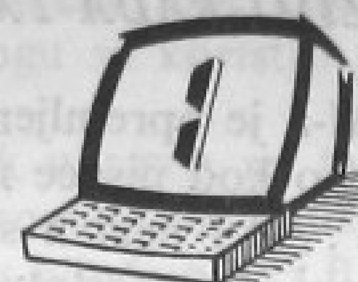
Uz svaku glavu dat je i veći broj primera — programa koji ilustruju obrađenu oblast tako da je uputno knjigu čitati pred ekranom TV prijemnika i uključenim SPECTRUM-om, pa svaki primer odmah odvežbati na računaru. Preporučljivo je ne prelaziti na sledeću oblast, dok se ne savladaju osnove gradiva izloženog u tekućoj.

Primeri i zadaci dati u knjizi različite su težine, pa pretpostavljamo da će biti interesantni i za početnike i za one koji se već bave programiranjem.

Posebna pažnja u knjizi posvećena je grafici i računarskim igrama. Obrađeno je definisanje grafičkih simbola, upravljanje njihovim pokretanjem i formiranje prostijih igara. U poslednjoj glavi date su četiri kompletne igre i objašnjen je njihov rad. Verujemo da će vam pružiti mnogo zadovoljstva.

Na kraju želimo da se zahvalimo svima koji su nam pomogli pri izradi ove knjige.

Beograd, decembar 1984. *Autori*



Puštanje računara u rad

Računar SINCLAIR ZX SPECTRUM sadrži sledeći pribor potreban za priključenje:

— ispravljač 220 V naizmeničnog napona na 9 V jednosmernog napona,

— koaksijalni kabl za priključenje na televizijski prijemnik,

— dvostruki kabl za povezivanje sa kasetofonom.

Računar se može priključiti na TV prijemnik koji ima mogućnost prijema UHF područja (drugi program). Takođe je potrebno da ima koaksijalni priključak za antenski ulaz. Ako to nije slučaj, potrebna je izrada adaptivnog priključka. Računar se može priključiti na TV prijemnik za prijem u boji ili za crno-beli prijem. U prvom slučaju bićete u mogućnosti da koristite osam boja kojima SPECTRUM raspolaže. Ako je vaš prijemnik crno-beli, "boje" će biti predstavljene raznim nijansama sive — između crne i bele boje.

Povezivanje računara i TV prijemnika vrši se koaksijalnim kablom koji se uključuje na priključak označen sa TV — na računaru, i sa UHF na antenski ulaz na TV prijemniku. Izlaz iz ispravljača (9 V) se uključuje na priključak 9 V DCin na računaru. Kada se ova povezivanja obave, može se priključiti ispravljač na 220 V i uključiti TV prijemnik.

Po uključenju TV prijemnika treba odabrati UHF područje i pomerati dugme za izbor kanala sve dok se na ekranu ne dobije poruka: c 1982 Sinclair Research Ltd.

1.1. REŽIMI RADA TASTATURE

Tastatura SPECTRUM-a je opremljena sa 40 dirki. Slova i brojevi su na istim mestima kao kod pisaće mašine. Gornji deo tastature sadrži cifre 1, 2, 3, . . . , 0 i naziva se numeričkim. Tri donja reda predstavljaju alfabetski deo tastature. Tu se nalaze slova A — Z. Ovde ne spadaju dirke ENTER, CAPS SHIFT, SYMBOL SHIFT koje imaju specifičnu upotrebu.

Pritiskom na neku od dirki na ekranu će se pojaviti odgovarajući znak, i to na poziciji na kojoj se pre pritiska nalazio kursor. Kursor služi za obeležavanje tekuće pozicije na ekranu. Da bi se razlikovao od ostalih znakova, uvek je dat u formi kontrasta u odnosu na tekuće boje ekrana i zapisa. Sadržaj kursora je slovo koje nas informiše u kom se režimu rada nalazi tastatura. On stalno trepće da bi se razlikovao od ostalih znakova na ekranu. Tako, ako se radi sa ekranom koji je bele boje i po kome se piše crnom, i ako se nalazimo u "L" režimu rada tastature, kursor će biti crn kvadrat u čijoj se unutrašnjosti nalazi slovo L ispisano belom bojom.

Karakteristika SPECTRUM-a je da se svakoj dirki dodeli nekoliko različitih namena. Tako, na primer, pritiskanjem na dirku H možemo dobiti:

— H (veliko slovo),
— h (malo slovo),
— naredbu GO SUB,
— simbol ↑ koji služi za obavljanje operacije stepenovanja,
— funkciju SQR koja služi za izračunavanje kvadratnog korena,
— naredbu CIRCLE kojom se mogu crtati krugovi.

Ovih šest različitih funkcija iste dirke ostvaruju se kroz različite režime rada tastature.

Postoji pet različitih režima rada tastature, i to :

— "K" (keyword) služi za ulaz ključnih reči, odnosno BASIC naredbi,
— "L" (letter) služi za ulaz slova (malih),
— "C" (capitale) služi za ulaz slova (velikih),
— "E" (extended) služi za ulaz BASIC naredbi, operatora i funkcija (njihovi nazivi su napisani ispod i iznad dirki na tastaturi)
— "G" (graphics) služi za ulaz grafičkih znakova.

Crveno označene funkcije, operatori i simboli koji se nalaze u desnom delu dirke mogu se koristiti u "K", "L" i "C" režimu rada.

Sadržaj kursora uvek ukazuje na tekući režim rada tastature. Tako, ako smo u grafičkom režimu rada, sadržaj kursora je "G", ako smo u režimu za unos ključnih reči BASIC-a, sadržaj je "K" itd.

Pozicioniranje u neki od režima rada tastature vrši se korišćenjem dirki CAPS SHIFT i SYMBOL SHIFT.

Prvi režim u kome će se tastatura naći po uključenju računara je "K" (keyword). To smo mogli i očekivati zato što po uključenju računar očekuje neku naredbu (tj. odgovor na pitanje — šta prvo po uključenju da uradi). Ako pritisnete bilo koju dirku, na ekranu će se pojaviti naredba ispisana belim slovima u dnu dirke. Tako dobijamo:

— dirka W, naredba DRAW
— dirka I, naredba INPUT
— dirka G, naredba GO TO
— dirka 1, broj 1.

Ako istovremeno pritisnemo dirku SYMBOL SHIFT i neku od dirki, na ekranu će se pojaviti sadržaj napisan crvenim slovima u desnom delu dirke. Tako je:

— dirka D, opcija STEP
— dirka W, operator <>
— dirka I, forma AT
— dirka 1, ! .

Navedeni simboli su ispisani na dirci crvenom bojom, a SYMBOL SHIFT je takođe "crven".

U "K" režimu rada tastature dirka CAPS SHIFT se koristi samo uz dirke sa numeričkog dela tastature. U tom slučaju ona aktivira funkcije napisane belim slovima iznad dirki. Tako je :

— dirka 8, → funkcija pomeranja kursora udesno,
— dirka 6, ↓ funkcija pomeranja kursora nadole.

"K" režim rada automatski smenjuje "L" režim rada u slučajevima kada je uneto THEN, ENTER, : (dve tačke), tj. u slučajevima kada se očekuje unos neke BASIC naredbe koja je ključna reč.

Ako se nalazite u "K" režimu, pritisnite na dirku P. Na ekranu će se pojaviti naredba PRINT koja vrši funkciju prikaza na ekranu. Uz ovu naredbu može da ide kao argument i tekst pod navodnicima kao:

PRINT "srecna Nova Godina!"

Posle naredbe PRINT, koja se već nalazi na ekranu, treba uneti (") — navodnik. Po onome što smo već rekli treba pritisnuti dirke SYMBOL SHIFT i P zajedno. Posle toga na ekranu je prikazano :

PRINT "

Posle unosa naredbe PRINT kursor se promenio sa "K" u "L". Ovaj prelaz u "L" režim rada tastature izvršen je automatski zato što u sklopu naredbe koju smo napisali računar očekuje da počnemo da unosimo neki tekst. Kucajmo tekst . . . srecna . . . Sada za unos svakog slova upotrebljavamo posebnu dirku. Blanko znak unosimo pritiskom na dirku SPACE. Sva slova koja je do sada trebalo uneti bila su mala. Slova N i G (Nova Godina) se unose u "L" režimu rada slično kao kod pisaće mašine. Treba istovremeno pritisnuti dirku CAPS SHIFT i željenu dirku.

Ako smo završili sa unosom celog teksta čestitke, na ekranu bi trebalo da piše :

PRINT "srecna Nova Godina

Da bi ova naredba bila korektna, potrebno je završiti sa (") navodnicima. Navodnik se nalazi na dirci P i označen je crvenom bojom. Ako pritisnemo na dirke P i SYMBOL SHIFT, dobićemo navodnik.

U "K" i "L" režimu rada tastature dirka SYMBOL SHIFT ima istu funkciju.

Ono što smo do sada uneli predstavlja naredbu. Kao rezultat te naredbe pojaviće se na ekranu tekst . . . srecna Nova Godina. Da bismo saopštili računaru da smo završili sa unosom naredbe, potrebno je pritisnuti dirku ENTER. Pritiskom na ovu dirku saopštavamo računaru da je završen unos naredbi, odnosno podataka.

Ako smo pritisnuli ENTER, dobićemo čestitku na ekranu, a kursor će opet biti u "K" režimu rada. To smo mogli i očekivati, jer posle unosa jedne naredbe računar očekuje drugu.

Unećemo sada naredbu INPUT koja služi za ulaz podataka.

INPUT "KOLIKO IMATE GODINA"; I

Kada unesemo INPUT i (") na isti način kao do sada dolazimo u "L" režim rada. Tekst koji se sastoji isključivo od velikih slova možemo uneti i na drugačiji način, a ne korišćenjem CAPS SHIFT pri kucanju svakog slova. To se postiže prelaskom u "C" režim.

Prelaz iz "L" režima u "C" režim i obratno obavlja se istovremenim pritiskanjem dirki CAPS SHIFT i 2 (CAPS LOCK). Kada pređemo u "C" režim, u kursoru se pojavljuje C.

Ako ste napisali celu naredbu, potrebno je saopštiti računaru da je naredba kompletirana i da počne sa njenim izvršenjem. Pritisnućemo ponovo dirku ENTER.

Za razliku od prošlog puta, sada ne dobijamo poruku pri vrhu ekrana, koju sledi promena kursora u "K", već nešto sasvim drugo. Pitanje KOLIKO IMATE GODINA se pojavljuje na dnu ekrana. Sada kursor ostaje pozicioniran iza navedenog teksta, i to u "C" režimu. To je zato što računar očekuje da unesemo neku brojčanu vrednost.

Kucajte 17 i posle toga pritisnite ENTER. Ovoga puta korišćenjem dirke ENTER saopštili smo računaru da smo završili sa unosom podataka.

Sada je kursor ponovo u stanju "K", tj. očekuje se od nas da unesemo neku novu naredbu.

U grafički režim rada tastature prelazi se istovremenim pritiskanjem dirki CAPS SHIFT i 9. Tada kursor ima oznaku G. Na tastaturi SPECTRUM-a postoji 8 fiksno definisanih grafičkih simbola koji se nalaze na dirkama 1—8. U ovome režimu rada koriste se i funkcije TRUE VIDEO i INV. VIDEO.

Svaki od 8 grafičkih simbola ima svoju inverznu sliku. Na dirci 5 nalazi se kvadrat čija je leva polovina bela, a desna je zatamnjena. Inverzna slika je kvadrat čija je leva polovina zatamnjena, a desna bela. Prelaz na korišćenje inverznih slika dobija se istovremenim pritiskom na CAPS SHIFT i dirku 4.

Prelaz na korišćenje normalnih likova grafičkih simbola vrši se istovremenim pritiskanjem na CAPS SHIFT i dirku 3. Iz grafičkog režima se izlazi pritiskom na dirku 9.

Postoji još jedan režim rada tastature u kome mogu da se koriste funkcije BASIC-a napisane crvenim odnosno zelenim slovima, a nalaze se iznad odnosno ispod dirki. To su u odnosu na dirku B funkcije BIN (zelena) i BRIGHT (crvena). One se mogu koristiti u "E" režimu rada tastature. U ovaj režim se prelazi istovremenim pritiskanjem na dirke CAPS SHIFT i SYMBOL SHIFT. Funkcije napisane sa gornje strane dirke (zeleno) dobiju se pritiskom na odgovarajuću dirku, a funkcije sa donje strane (crvene) istovremenim pritiskom na odgovarajuću dirku i na jednu CAPS

dirku (bilo koju). U ovom režimu rada se ostaje samo za vreme jednog unosa funkcije.

Pri kucanju se često dešava da napravimo grešku. Ako, na primer, unesemo:

PRINT PBEOGRAD"

umesto:

PRINT "BEOGRAD"

zato što smo kod unosa (") zaboravili da pritisnemo dirku SYMBOL SHIFT, posle pritiska na dirku ENTER računar će signalizirati da smo napravili grešku. To čini treperućim upitnikom "?". U tom slučaju naredba koju smo napisali nije prihvaćena od strane računara.

Ispravka se može izvršiti na sledeći način:

— pritiskanjem na ← (dirke CAPS SHIFT i 5) pomeramo kursor sve dok ne dođemo do pozicije za koju mislimo da je pogrešna,

— pritiskamo DELETE (dirke CAPS SHIFT i 0), čime brišemo poziciju koja neposredno prethodi kursoru,

— unosimo ispravnu vrednost — u ovom slučaju to je ("),

— pomeramo kursor udesno pritiskanjem na → (dirke CAPS SHIFT i 8) sve dok kursor ne dođe do kraja naredbe koja se ispravlja.

Ako se sada pritisne ENTER, naredba će biti prihvaćena i na ekranu će se ispisati BEOGRAD.

Unesimo sada jedan jednostavan program koji se sastoji samo od nekoliko linija.

	10	LET		a		=	3	
"K"	10	L	"L"	A	SS	L	3	ENTER
	20	LET		b		=	4	
"K"	20	L	"L"	B	SS	L	4	ENTER
	30	PRINT		a		+	b	
"K"	30	P	"L"	A	SS	K	B	ENTER
	RUN	ENTER						
"K"	R							

SS označava pritiskanje na dirku SYMBOL SHIFT, a "K" i "L" odgovarajuće režime rada tastature. RUN je komanda kojom se pokreće izvršenje programa. Kao rezultat dobićemo 7.

Pošto unesemo neku naredbu i pritisnemo dirku ENTER, prihvaćena naredba će se pojaviti na gornjem delu ekrana. Uz poslednju prihvaćenu liniju, u gornjem delu ekrana, pojavljuje se znak ">". Na taj način znamo koja je linija takozvana tekuća linija. Znak ">" nazivamo programski kursor. On se može pomerati dole-gore korišćenjem dirki 6 i 7 zajedno sa CAPS SHIFT dirkom. To se praktikuje kada želimo da menjamo neku već unetu, tj. prihvaćenu programsku liniju.

Promenimo broj 4 u liniji broj 20 u 17. Postupak je sledeći:

— uz pomoć dirke 6 ili 7 (pritisnute zajedno sa CAPS SHIFT dirkom) pomeramo programski kursor na liniju broj 20,

— pritisne se EDIT (zajedno dirke CAPS SHIFT i 1); posle toga linija broj 20 se pojavljuje i u dnu ekrana,

— uz pomoć → (dirke 8 i CAPS SHIFT zajedno) pomeramo "L" kursor sve dok se ne dođe do kraja linije,

— potom uz pomoć funkcije DELETE brišemo 4 i upisujemo 17,

— kada se promena sprovede pritiska se ENTER.

Time je sadržaj linije 20 promenjen. Ako sada damo naredbu RUN, dobićemo novi rezultat 20.

1.2. KORIŠĆENJE KASETOFONA

Ako imamo neki program na kaseti i želimo da ga koristimo za učitavanje programa u memoriju SPECTRUM-a, koristićemo se naredbom LOAD. Potrebno je pre toga naredbom NEW obrisati sadržaj memorije.

Da bismo priključili kasetofon na računar potrebno je, prvo, priključiti jedan kraj dvostrukog kabla na ulaz u SPECTRUM označen sa EAR, a drugi kraj istog kabla na ulaz kasetofona označen najčešće sa LINE OUT. Oba kraja drugog kabla ostavimo da slobodno vise u vazduhu. Potenciometar kasetofona treba podesiti na oko 1/4 jačine zvuka. Potenciometar za boju zvuka treba podesiti na niske tonove. Potom treba tipkati:

LOAD ""

posle čega ćete pritisnuti dirku ENTER. Tada treba startovati reprodukciju na kasetofonu. Na ekranu prijemnika prvo će se menjati crveni i plavi spoljni deo, a zatim će se pojaviti naizmenično crvene i plave pruge širine oko 1 cm. Kada računar počne da učitava program, preko ekrana će se pojaviti tanke žuto-plave linije, a naziv programa će biti ispisan na ekranu.

Ako se na ekranu ne pojave opisani efekti, treba pojačavati jačinu zvuka do oko 1/2. Ako i tada program ne može da se učita, treba pomerati potenciometar za boju zvuka prema visokim tonovima, a zatim opet pojačavati jačinu zvuka sve dok se program ne učita.

Ako program ne budemo mogli da učitamo, možemo koristiti izlaz za eksterni zvučnik, ali tu moramo biti veoma obazrivi sa jačinom zvuka.

Na isti način treba da postupimo ako nam pri učitavanju računar odgovori porukom:

R Tape Loading Error

Kada je program učitan, startovaćemo njegovo izvršavanje naredbom RUN. Napomenimo da se program može upisati na kasetu tako da se odmah po učitavanju pokrene njegovo izvršenje.

Izbor određenog programa sa kasete možemo vršiti naredbom:

LOAD "ime programa"

S druge strane forma LOAD "" učitava prvi program na koji se naiđe počev od tekuće pozicije kasete.

Ako na kaseti imamo program koji je pisan u mašinskom jeziku, učitavanje se vrši sa:

LOAD "" CODE

Za prvo učitavanje programa dobro će nam poslužiti demonstraciona kaseta koju smo dobili kada smo kupili SPECTRUM. Na njoj se nalazi program SEIDA snimljen veći broj puta, tako da, dok isprobavate učitavanje menjajući jačinu zvuka i boju, ne morate stalno premotavati kasetu na početak.

Pogledajmo sada postupak upisivanja programa, koji se nalazi u memoriji računara, na kasetu.

Tu ćemo koristiti i druga dva utikača dvostrukog kabla. U priključak MIC na računaru stavićemo jedan, a drugi utikač stavićemo u ulaz za mikrofon (LINE INPUT) na kasetofonu.

U početku treba isključiti kabl, koji smo koristili, da nam se ne bi pojavio povratni šum, i to sve dotle dok se ne ustanovi da li nam držanje uključenog kabla za čitanje ne stvara probleme. Stavimo praznu kasetu na koju ćemo snimati program i tipkajmo:

SAVE "ime programa"

Računar će nam odgovoriti sa:

Start tape, then press any key

Zabeležićemo stanje na brojaču kasetofona da bismo znali gde počinje program, postavićemo kasetofon na snimanje, a zatim pritisnuti bilo koju dirku.

Kada se upis programa završi, na ekranu ćemo dobiti poruku:

0 OK, 0 : 1

Pošto uvek postoji mogućnost da se pri snimanju jave greške, preporučuje se da se proveri da li snimak odgovara originalu, tj. da li program sa kasete odgovara programu u memoriji SPECTRUM-a.

Taj postupak se obavlja na sledeći način. Prvo treba premotati kasetu na poziciju koja odgovara početku programa. Tu poziciju smo zabeležili kada smo počinjali snimanje programa. Zatim ćemo priključiti kabl za čitanje programa na isti način kao što smo to već radili (kada je opisivana naredba LOAD). Treba isključiti kabl za snimanje i startovati izvršenje naredbe VERIFAY "". Potom treba startovati reprodukciju i sačekati da se program koji je upisan na kaseti uporedi sa programom koji je u memoriji.

Ako dobijemo poruku:

0 OK, 0 : 1

možemo biti sigurni da je snimanje potpuno uspelo i da ćemo ga sledeći put bez problema učitati.

U slučaju da nam računar odgovori porukom:

R Tape loading error, 0 : 1

postupak upisa i verifikacije programa moramo ponoviti.

Već smo rekli da se program može tako zapisati da odmah po njegovom učitavanju počne postupak izvršenja. To možemo da ostvarimo sledećom formom naredbe SAVE :

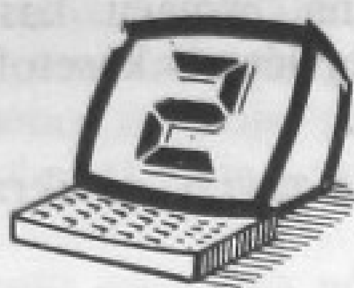
SAVE " ime programa " LINE broj

Ovako zapisan program počeće po učitavanju odmah sa izvršenjem, i to počev od naredbe koja se nalazi na liniji pod navedenim brojem.

Primer :

SAVE "PROGRAM 1" LINE 30

Ovako zapisan program odmah po učitavanju počeće da se izvršava i to počev od naredbe koja se nalazi na liniji sa brojem naredbe 30.



Prvi program

2.1. NAREDBA PRINT

PRINT je svakako jedna od naredbi BASIC-a koja se najviše koristi. Ovu naredbu koristimo kada želimo da računar nešto odštampa (prikaže) na ekranu televizora. Unesite sledeću naredbu i posle pritisnite dirku ENTER:

PRINT 2

Po izvršenim operacijama računar će prikazati 2 na ekranu.

Računar se može koristiti i kao džepni kalkulator. Ako unesemo sledeće naredbe:

PRINT 1+1
PRINT 256*17

pri čemu ćemo posle unosa svake pritiskati ENTER, dobićemo na ekranu rezultat sabiranja (2), odnosno množenja (4352). Zapazimo da je u BASIC-u* operator koji se koristi kada se vrši operacija množenja.

Rad u tzv. direktnom načinu rada omogućava izvođenje nekad i veoma složenih računanja. Ispitajte to na primeru koji sledi. Ne zaboravite da pritisnete ENTER pošto unesete sledeće naredbe:

PRINT SQR (5+4)	rezultat je	3
PRINT SQR (3↑2+4↑2)		5

Prva naredba traži od računara da prikaže na ekranu kvadratni koren zbira 5 i 4. Simbol SQR ima značenje kvadratnog korena i jedna je od nekoliko matematičkih funkcija kojima računar raspolaže.

Druga naredba računa kvadratni koren — funkcija SQR, zbira kvadrata 3 i 4. Ako želimo da izvršimo stepenovanje, koristićemo funkciju ↑. Tako u našem primeru 3↑2 znači 3^2, a 4↑2 znači 4^2.

Naredbom PRINT možemo na ekranu ispisivati i tekstove. Pritisnite istovremeno CAPS SHIFT i dirku 2, tj. prenesite klavijaturu u CAPS LOCK stanje. Možemo reći da se on sada ponaša kao pisaća mašina, i to spremna za kucanje velikih slova. Unesite sledeću naredbu i, naravno, ne zaboravite da pritisnete ENTER.

PRINT DOBAR DAN, KAKO STE

Na žalost, umesto da nam se na ekranu pojavi ovo ljubazno pitanje, pojaviće se poruka da smo napravili grešku. Na ekranu će se pojaviti tekst: ERROR 2, što se tumači kao pojava nedefinisane promenljive u okviru naredbe čije smo izvršenje želeli. Greška koju smo napravili je sledeća: ako na ekranu želimo da prikažemo neki tekst, moramo ga staviti među navodnike. Uradimo dakle:

PRINT "DOBAR DAN, KAKO STE"

a posle toga, naravno, pritisnućemo ENTER i dobićemo na ekranu pitanje:

DOBAR DAN, KAKO STE

Na žalost, još uvek ne znamo kako možemo da odgovorimo na ovo ljubazno pitanje, pa se za sada potrudimo da zapamtimo ono najvažnije, da svaki tekst koji želimo da prikažemo na ekranu treba da stavimo među navodnike. Navodnik (") ne mešajte sa apostrofom ('), jer oni u BASIC-u imaju sasvim različita značenja.

Ako bismo želeli da prikažemo na ekranu:

PRINT "IZDAVACKO PREDUZECE" TEHNICKA KNJIGA" BEOGRAD"

— nećemo uspeti. Ovako napisanu naredbu računar ne prihvata. Ova greška nastaje zbog toga što se u okviru teksta koji želimo da prikažemo na ekranu javlja i navodnik ("). Sa druge strane, videli smo da navodnik ima funkciju označavanja početka i kraja teksta. Taj problem se rešava tako što, ako se u okviru teksta pojavi navodnik kao deo teksta, on se piše dva puta. Prema tome unesite:

PRINT "IZDAVACKO PREDUZECE" "TEHNICKA KNJIGA" "BEOGRAD"

i posle toga, naravno, pritisinite ENTER. Videćemo da se na ekranu pojavljuje ono što smo želeli.

Posle ovih početnih koraka koje smo načinili u radu sa računarom, pokušajmo da uradimo jedan program, naravno veoma lak. Program je bez jedne od naredbi PRINT.

Unesite sledeću sekvencu naredbi. Svakako, više nema potrebe da se podsećamo da posle svake unete naredbe, u ovom slučaju posle svakoga reda, pritisnemo dirku ENTER. To je nešto što smo već naučili i na šta se više nećemo podsećati.

```
10 REM      PRVI PROGRAM
20 PRINT    "OVO JE PRVI PROGRAM"
30 PRINT    "PRVI SABIRAK JE"; 3
40 PRINT    "DRUGI SABIRAK JE"; 5
50 PRINT    "ZBIR JE"; 3+5
60 PRINT    "OVO JE KRAJ PROGRAMA"
```

Da bismo izvršili ovaj program, treba pritisnuti RUN i posle toga ENTER. Na ekranu će se pojaviti rezultat rada programa i to:

```
OVO JE PRVI PROGRAM
PRVI SABIRAK JE   3
DRUGI SABIRAK JE  5
ZBIR JE           8
OVO JE KRAJ PROGRAMA
```

Pogledajmo šta ima novo za nas u ovom programu. Prva linija koja je numerisana sa 10 sadrži naredbu REM (skraćenica od engl. REMARK, tj. primedba, komentar). Naredba REM nema nikakvog uticaja na izvršenje programa. Njena jedina uloga je pojašnjenje rada programa. Tekst koji sledi REM može sadržati sve znake — numeriku, alfanumeriku, interpunkciju i blanko. Iako su ovi komentari veoma korisni, pogotovo kada se vrše ispravke programa ili analiza tuđih programa, sa njihovom upotrebom ne treba preterivati zato što, iako ne utiču na izvršenje, oni zauzimaju memoriju.

REM se često sreće u formi:

```
10 REM RESENJA KVADRATNE JEDNACINE
2680 REM POMERANJE RAKETA
```

U slučaju da nam je to potrebno, možemo da koristimo nekoliko uzastopnih naredbi REM, tj. nekoliko linija teksta komentara, na primer:

```
120 REM    OVAJ PODPROGRAM
130 REM    VRSI VERTIKALNO I
140 REM    HORIZONTALNO POMERANJE RAKETE
```


ili:

```
4729 REM    ****************************
4730 REM    *  PRELAZ U DRUGU SOBU
4731 REM    ****************************
```

Drugo, što smo takođe primetili u programu, jeste to da ispred svake naredbe stoji broj (u ovom slučaju od 10—60). Broj naredbe može uzimati vrednost od 1 do 9999. Broj naredbe je jedinstven u okviru programa, te shodno tome jedinstveno određuje neku naredbu u okviru programa. Videćemo kasnije, kada budemo želeli da se pozovemo na neku naredbu, činićemo to pozivanjem na broj naredbe. Brojevi naredbi se najčešće daju sa proredom da bi kasnije lakše mogle da se vrše izmene programa, tj. da se ubacuju nove linije, itd.

Pokušajmo sada da malo izmenimo naš program. Prvo ćemo ga prikazati na ekranu. To se radi naredbom LIST. Prema tome, pritisnite LIST, a posle toga ENTER. Slika programa koja se pojavila na ekranu najčešće se naziva LISTING PROGRAMA. Unosimo sada broj 30 i pritisnemo ENTER. Linija koja je bila numerisana sa 30 je nestala. Znači, da bismo obrisali neku liniju dovoljno je napisati broj linije (naredbe) i posle toga pritisnuti ENTER. Sada unesite naredbu:

```
30 PRINT "PRVI SABIRAK JE"; 12
```

Pritisnite LIST ponovo i dobićemo novi listing programa. Vidimo da je stara linija 30 zamenjena novom. Primetićemo da naredbe u programu uvek stoje u redosledu brojeva naredbi, odnosno linija.

Ako želimo da unesemo novu liniju, postupimo na sledeći način. Unesimo naredbu:

```
15 REM SA IZMENAMA
```

i dajmo LIST;

Vidimo da na listingu programa postoji i nova linija. Sada znamo kako da brišemo, menjamo i dodajemo nove linije u program, odnosno da menjamo program. Na ekranu sada dobijamo:

```
10 REM PRVI PROGRAM
15 REM SA IZMENAMA
20 PRINT "OVO JE PRVI PROGRAM"
30 PRINT "PRVI SABIRAK JE"; 12
40 PRINT "DRUGI SABIRAK JE"; 5
50 PRINT "ZBIR JE" ; 3+5
60 PRINT "OVO JE KRAJ PROGRAMA"
```

Pokrenimo sada izvršenje ovako izmenjenog programa. Već znamo kako se to radi. Pritisnite RUN, a zatim ENTER. Računar će da izvrši naredbu po naredbu po redosledu kako je to dato na listingu programa. Kao rezultat izvršenja dobili smo sledeću sliku na ekranu :

```
OVO JE PRVI PROGRAM
PRVI SABIRAK JE  12
DRUGI SABIRAK JE  5
ZBIR JE          8
OVO JE KRAJ PROGRAMA
```

Dobili smo nešto što će nas svakako začuditi. 12+5=8 ? Izgleda da je računar pogrešio. Ako malo pažljivije pogledamo, videćemo da to nije tačno. Računari veoma retko greše! Ako smo dobro upoznali kako radi, možemo čak reći nikada. Sve greške nastaju zbog toga što smo mu loše formulisali ono što treba da uradi, odnosno loše smo programirali. Podsetimo se šta smo do sada sve radili sa prvim programom.

— Napisali smo ga i pozvali na izvršenje. Rezultat sabiranja bio je dobar.

— Izmenili smo naredbu 30 u 30 PRINT "PRVI SABIRAK JE" ; 12

— I dodali smo naredbu novu 15 REM SA IZMENAMA.

Znamo da naredba REM ne utiče na izvršenje, pa prema tome to i nije uzrok greške. Izmenom u naredbi 30 promenili smo vrednost prvog sabirka sa 3 u 12. Ta promena ni na koji način ne utiče na izvršenje naredbe 50, gde se štampa zbir dve konstante 3 i 5. Prema tome, morali smo da izmenimo i vrednost prvog sabirka u redu 50.

Iako je PRVI PROGRAM bio veoma jednostavan, zapamtite nešto što je veoma važno u programiranju. Kada vršite neku izmenu u programu — uvek imajte na umu da se ta izmena mora uklopiti u program kao celinu, a ne samo u neki njegov deo.

Da početak upoznavanja sa programiranjem na SPECTRUM-u završimo sa naredbom NEW. Tom naredbom se briše sadržaj korisničke memorije. Ona mora biti korišćena pre unosa novog programa. Ako to ne bismo uradili, po unosu sledećeg programa dobili bismo umesto novog programa nešto što predstavlja kombinaciju prvog i drugog. Taj "treći program" bi svakako imao interesantan listing, ali teško da bi mogao da se izvršava. To je posledica onoga što smo već rekli o upotrebi brojeva naredbi pri izmeni

programa. Nemojmo još pritiskati NEW, već pokušajmo da unesemo sledeći program :

```
10 REM PROGRAM DVA
30 PRINT SQR (25↑3+2↑6)
50 REM KRAJ PROGRAMA DVA
```

Dajte sada LIST — dobićemo na ekranu "prvi program":

```
10 REM PROGRAM DVA
15 REM SA IZMENAMA
20 PRINT "OVO JE PRVI PROGRAM"
30 PRINT SQR (25↑3+2↑6)
40 PRINT "DRUGI SABIRAK JE"; 5
50 REM KRAJ PROGRAMA DVA
60 PRINT "OVO JE KRAJ PROGRAMA"
```

Računar opet nije pogrešio! Kada smo unosili drugi program, on je to shvatio (zato što nismo pritisnuli NEW) kao da želimo da izvršimo izmene prvog programa. Zato je i izvršio zamenu linija 10, 30 i 50 novim. Pritisnimo sad NEW. Posle toga možemo pritisnuti LIST i na ekranu nećemo dobiti listing.

2.2. PRINT SEPARATORI, OBLICI PRINT TAB — PRINT AT

Veoma često želimo da rezultati rada našeg programa, odnosno njihov prikaz na ekranu ima bogatiji izgled, tj. da rezultate predstavimo u formi neke tabele, grafika itd.

Jedna od mogućnosti za tabuliranje koju nam pruža SPECTRUM je i upotreba PRINT separatora (;), (,), ('). Pogledajmo, prvo, koja je njihova osnovna funkcija :

(;) tačka-zarez čini da broj ili tekst koji se štampa neposredno sledi prethodni,

(,) zarez čini da se štampa ili na levoj margini ili od sredine ekrana, a to zavisi od prethodnog pozicioniranja na ekranu u okviru prethodne ili iste naredbe PRINT,

(') apostrof čini da se ono što ga sledi štampa na početku novog reda.

Pritisnimo CAPS LOCK i unesimo sledeći program koji treba da nam demonstrira upotrebu (;) (,) (').

```
10 REM UPOTREBA ( ; ) ( , ) ( ' )
20 PRINT "PRVI RED" ; "A=" ; 10 ; "B=" ; 20
30 PRINT "DRUGI RED" ;"   A=" ;10 ;"    B=" ; 20
```

```
40 PRINT "TRECI RED"
50 PRINT "CETVRTI RED","A=" ; 10 ; " B=" ; 20
60 PRINT "PETI RED" ' "A=" ; 10, "B=" ; 20
70 PRINT "SESTI RED", "A=", 10, "B=", 20
80 PRINT
90 PRINT
100 PRINT "KRAJ PROGRAMA"
```

Izvršimo ovaj program i komentarišimo ga liniju po liniju:

10 — REM naredba koja daje naslov programa,

20 — 100 su PRINT naredbe,

30 — Prouzrokuje ispisivanje PRVI REDA= 10B=20, i to počev od leve margine ekrana. U ovom redu smo koristili samo (;). Primećujemo da se podaci koji se prikazuju na ekranu nalaze neposredno jedan iza drugog. Pošto to najčešće ne želimo, dodali smo u tekst po blanko znak i time razdvojili elemente prikaza na ekranu. To se vidi u redu 30.

40 — Prouzrokuje izlaz teksta CETVRTI RED, i to u posebnom redu. Primetimo da svaka naredba PRINT počinje od novog reda. Prema tome, možemo naredbu PRINT praćenu blanko znakom koristiti za prored među redovima. To koristimo kod naredbi 80 i 90.

50 — Ovde nam se prvi put pojavljuje (,). Vidimo da tekst CETVRTI RED počinje od leve margine ekrana, dok deo koji sledi (,) A=10 B=20 počinje od sredine ekrana.

60 — Ovde se koristi ('). Tekst PETI RED počinje na levoj margini ekrana. Pošto ga sledi apostrof, A=10 izlazi u sledećem redu, i to počev od leve margine, a B=20 počev od sredine ekrana.

70 — Tekst SESTI RED počinje od leve margine, A= od sredine ekrana u istom redu. U sledećem redu se pojavljuje 10 od početka, a B = od sredine ekrana. Broj 20 se pojavljuje od početka sledećeg reda. Primetimo da sadržaj koji sledi naredbu PRINT ide u ovom slučaju u tri reda.

Prema tome da li će se sadržaj koji sledi naredbu PRINT pojaviti u jednom ili više redova zavisi od vrste i rasporeda separatora (;)(,)(') koji mu pripadaju.

Videli smo da separatori (;)(,)(') vrše tzv. relativni pomeraj, odnosno pomeraj u odnosu na prethodno odštampano.

Uz naredbu PRINT možemo koristiti i opciju AT, čime možemo tačno odrediti poziciju na ekranu u kojoj želimo nešto da prikažemo. Ako, na primer, kažemo PRINT AT 10, 15; "*" prikazaće se zvezdica, i to u 11 redu i 16-oj koloni. Argumenti koji idu uz AT su red i kolona. Ekran ima 24 reda, tj. horizontalne pozicije, i 32 kolone, tj. vertikalne pozicije. Redovi su numerisani od 0—23, a kolone od 0—31, i to odozgo nadole i sleva nadesno. Prema tome, prvi red je numerisan sa nulom, drugi sa 1, treći sa 2 itd. Isto važi i za kolone. Naredbom PRINT ne možemo se pozicionirati na poslednje dve linije ekrana (tj. 22 i 23) zato što su one rezervisane za komande, ulaz podataka, prikaz nekih poruka i sl. Zato, kada govorimo o naredbi PRINT, pod zadnjom linijom podrazumevamo liniju 21.

Unesite i izvršite sledeći program :

```
10 REM ZVEZDA
20 PRINT AT  8, 16 ; "*"
30 PRINT AT  9, 14 ; "* * *"
40 PRINT AT 10, 15 ; "***"
50 PRINT AT 11, 13 ; "*******"
60 PRINT AT 12, 15 ; "***"
70 PRINT AT 13, 14 ; "***"
80 PRINT AT 14, 16 ; "*"
```

Dobili smo zvezdu koja je pozicionirana na sredinu ekrana.

Dok AT omogućava da se neki tekst prikaže na određenoj poziciji ekrana, dotle TAB vrši tabulaciju teksta. Uz TAB ide kao argument broj kolone u kojoj počinje da se štampa neki tekst.

Ako bismo želeli da napravimo program koji za neke vrednosti argumenta štampa vrednosti funkcije i ako želimo tome dati neku podesnu formu, možemo koristiti TAB.

Unesite i izvršite sledeći program.

```
10 REM TABULIRANJE
20 PRINT TAB 5 ; "POLUPRECNIK" ; TAB 17 ;
   "POVRSINA"
30 PRINT TAB 10 ; 10 ; TAB 17 ; 10↑2*PI
```

Na kraju unesite i sledeći program. Ovaj program je takođe demonstracija korišćenja naredbe TAB. On sadrži i mnoge elemente koji su nam za sada nepoznati. Zato je moguće da vam i pored objašnjenja ovaj program ne bude sasvim jasan. Zato, kada naučite i ostale elemente iz kojih je on sastavljen, vratite se ponovo na njega i neka vam za sada služi samo za zabavu. Pažnja, pre nego što počnete da ga unosite pritisnite dirku NEW!

```
 10 REM JAHAC
 20 DIM A$(5,5)
 25 BORDER 0
 30 FOR J=10 TO 1 STEP -1
 40 PRINT TAB 3*J; INK 2;J
 50 FOR A=1 TO J
 52 BEEP .02,5*J
 60 NEXT A
 70 NEXT J
 71 FOR B=1 TO 5
 72 READ Q$
 73 LET A$(B)=Q$
 74 NEXT B
 90 LET Q=INT (RND*25)+1
100 FOR R=1 TO 5
110 BEEP .02,10*R
120 PRINT "(";TAB Q;INK R;A$(R
); INK 0;TAB 30;")"
130 NEXT R
140 POKE 23692,-1
150 LET SPACE=Q/3
180 FOR P=1 TO SPACE
190 PRINT "(";TAB 30;")"
200 NEXT P
210 GO TO 90
220 DATA "  █ ","▄▄█ "," █ █
","▀▀▀ "," █ █ "
```

Sl. 2.1 — Program "Jahač"

Za nas su najinteresantnije naredbe programa koje se nalaze u linijama 120 i 190 zato što one koriste TAB opciju naredbe PRINT.

Naredbom 120 vrši se:

prikaz otvorene zagrade "(" na levoj margini ekrana.

TAB q vrši prikaz na q-toj poziciji ekrana figure čiji je izgled definisan naredbom 222 DATA. q može uzimati vrednosti 1—25, što uzrokuje pomeranje figure na ekranu ulevo i udesno.

INK određuje boju delova figure.

A$(r) određuje koji deo figure treba da se prikaže. Figura se sastoji iz pet delova i nalazi se u pet redova. Naredbe 71—74 predstavljaju tzv. programsku petlju. Tim naredbama se vrši formiranje figure.

INK 0 određuje crnu boju

TAB 30 vrši prikaz na poziciji 30 zatvorene zagrade ")".

Naredbom 190 vrši se prikaz otvorene zagrade "(" na levoj margini ekrana i zatvorene ")" na poziciji 30 ekrana. Ova se naredba koristi kada se ne vrši prikaz figure.

SIMBOL	KOD	Kako se dobija	SIMBOL	KOD	Kako se dobija
	128	G 8		133	G 5
	143	G 8 shift		138	G 5 shift
	140	G 3 shift		139	G 4 shift
	136	G 7 shift		135	G 7

Sl. 2.2 — Grafički simboli korišćeni u programu "Jahač", njihovi kodovi i način dobijanja

Izvođenje ovog programa možete da prekinete pritiskom na dirku BREAK.

Podsetimo se, ako želite da sačuvate ovaj program na kaseti vašeg magnetofona, to možete učiniti uz pomoć naredbe SAVE, iza koje se piše naziv programa između navodnika, kao na primer:

SAVE "PROGRAM JEDAN"

SAVE "IGRA"

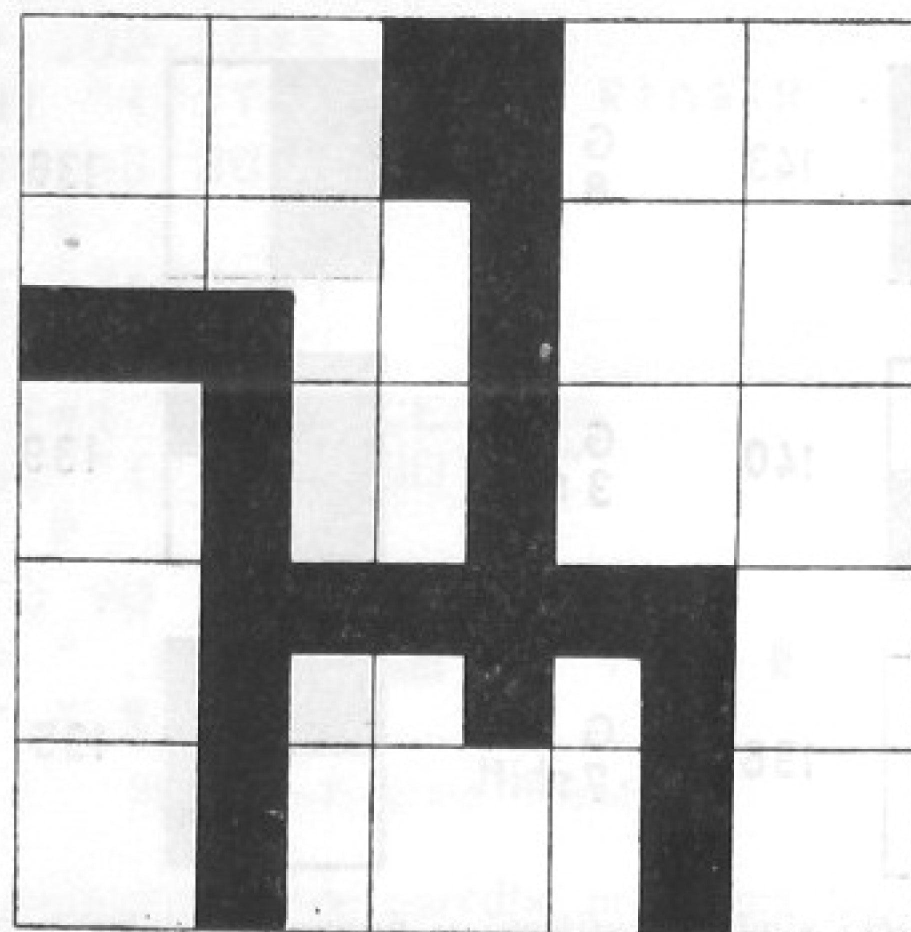
Ako se odlučimo da ovaj program zovemo „JAHAC", uradićemo :

SAVE "JAHAC"

posle čega ćemo pritisnuti dirku ENTER. Na ekranu će se pojaviti poruka :

START TAPE THEN PRESS ANY KEY

što znači: startujte vaš magnetofon (u stanju spremnom za snimanje), a potom pritisnite bilo koju dirku na tastaturi računara. To ćemo i učiniti, pozicioniraćemo kasetu na poziciju gde nema drugog zapisa, i tu poziciju zapamtiti, pritisnuti dirke PLAY/REC na magnetofonu, podesiti jačinu zvuka i potom pritisnuti bilo koju dirku na tastaturi računara.



Sl. 2.3 — Figura "Jahač"

Kada se upisivanje na kasetu završi dobićemo poruku:

0 OK, 0 : 1

Tada treba zaustaviti kasetofon i, naravno, zabeležiti poziciju gde je kraj programa.



Obrada promenljivih

3.1. PROMENLJIVE

Postoje dve vrste promenljivih: numeričke i nenumeričke (ili STRING).

Numeričke promenljive kao svoj sadržaj mogu imati samo numeričke vrednosti. Za njih je karakteristično da je zapis dat u formi pokretnog zareza, odnosno posebno se upisuje mantisa, a posebno eksponent. Područje za memorisanje je dugačko pet okteta, što dozvoljava da se unose vrednosti u intervalu $\pm 3*10^{-39} \pm 7*10^{+38}$. Preciznost zapisa ide do 9 značajnih cifara. Naziv numeričke promenljive može se sastojati od slova i brojeva, pri čemu prvi znak mora da bude slovo. Blanko znakovi u nazivu promenljive se ignorišu, a podjednako se tretiraju velika i mala slova. Pogledajmo nekoliko primera pravilno i nepravilno napisanih numeričkih promenljivih.

Ispravno

A

A1

FUNKCIJA BROJ 1

PRVI KOREN

PRVIKOREN

Neispravno

20	počinje sa brojem
2A	počinje sa brojem
FUNKCIJA—BROJ—1	sadrži —, odnosno znak koji nije ni slovo ni broj.

Ne postoji ograničenje u pogledu dužine naziva promenljive, ali iz praktičnih razloga nikad ne uzimamo previše duge nazive (ide se najčešće do 5 — 6 znakova). Takođe je poželjno da naziv promenljive asocira na njen sadržaj, jer će nam za vreme izrade programa biti mnogo lakše da radimo sa promenljivim koje se zovu BRZINA, VREME, MASA itd. nego sa xl, y, B3f, A861 itd.

Od ovoga što smo naveli postoji jedan izuzetak. Numerička promenljiva koja se koristi u naredbi FOR može imati naziv dužine 1, tj. može biti samo slovo.

Unesite i pozovite na izvršenje sledeći program:

```
10 REM OBRACUN KAMATE PO STOPI 7,5
20 PRINT "OBRACUN KAMATE PO STOPI 7,5"
30 LET IZNOS = 1823
40 PRINT IZNOS
50 LET IZNOS = IZNOS * 1.075
60 GO TO 40
```

Nove naredbe u ovom programu su LET i GO TO. Naredba LET služi za dodeljivanje vrednosti nekoj promenljivoj. Ako, na primer, kažemo LET A = 0, to znači da će sadržaj ove promenljive posle izvršenja naredbe postati 0 bez obzira na vrednost koju je promenljiva A do tada imala.

Unesite i pozovite na izvršenje sledeći program:

```
10 REM ILUSTRACIJA RADA LET NAREDBE
20 LET A = 10
30 PRINT A
40 LET A = 20
50 PRINT A
60 LET B = 35
70 PRINT A+B
80 LET C = A+B
90 PRINT C
100 LET A = A*B
110 PRINT A
```


Na ekranu će jedna za drugom biti prikazane vrednosti (10, 20, 55, 55, 700). Primetimo da se sa desne strane znaka jednakosti može naći i aritmetički izraz, a ne samo konstanta ili promenljiva. To je slučaj kod naredbe 80 LET C = A+B. Značenje naredbe je sledeće: "saberi sadržaj promenljive A sa sadržajem promenljive B i dobijeni zbir dodeli promenljivoj C. Sadržaj promenljivih A i B se ne menja." Primetimo da prethodni sadržaj promenljive C ne utiče na novu vrednost.

Vratimo se ponovo programu računanja kamata. Pogledajmo naredbu 50 LET IZNOS = IZNOS * 1.075. Ovde se promenljiva IZNOS pojavljuje i sa leve i sa desne strane znaka jednakosti (=). Kao i ranije, značenje naredbe je "pomnoži sadržaj promenljive IZNOS sa konstantom 1.075 i dobijeni proizvod dodeli promenljivoj IZNOS". Pri ovome promenljiva IZNOS menja svoj sadržaj i nova vrednost zavisi od stare. To je posledica toga što se promenljiva IZNOS nalazi sa obe strane znaka jednakosti. Slična ovome je sledeća naredba:

LET X=X+1

Po izvršenju ove operacije sadržaj promenljive X biće uvećan za 1, odnosno:

Sadržaj pre naredbe	*Posle naredbe*
10	11
26	27

LET IZNOS = IZNOS * 5 + 10

Sadržaj pre naredbe	*Posle naredbe*
100	510
—20	—90

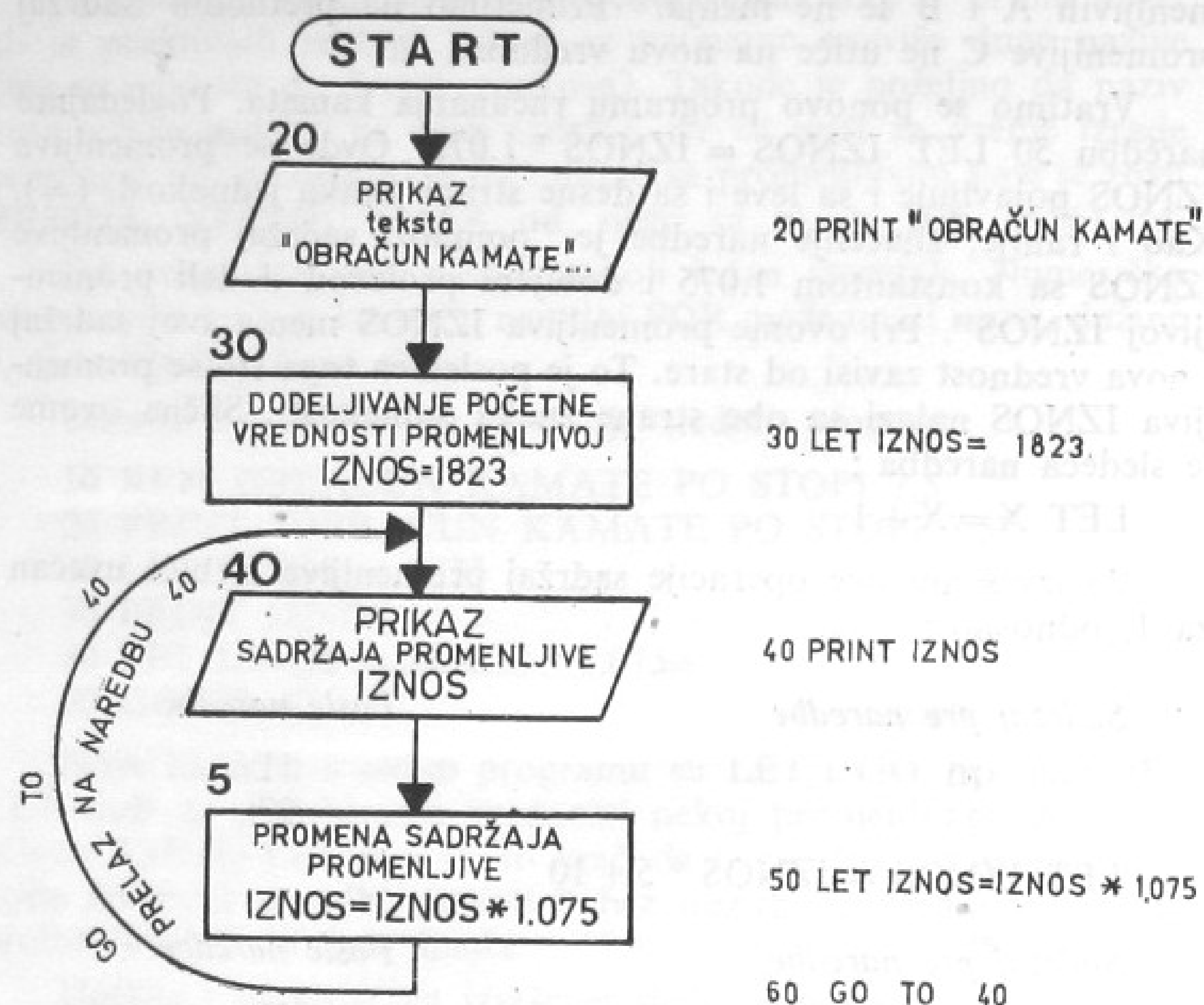
Sada je jasno kako se računa kamata:

(I) Prvom naredbom LET se dodeljuje promenljivoj iznos vrednosti 1823, što možemo da smatramo za vrednost uloga. Potom se ta vrednost (naredba PRINT) prikazuje na ekranu.

(II) Iznos se uvećava za iznos kamate (tj. za 7,5%).

Operacija (II) se ponavlja više puta. Svaki sledeći red predstavlja vrednost kamate u sledećoj godini. Ponavljanje operacije izračunavanja kamate je omogućeno uvođenjem naredbe GO TO 40 (idi na naredbu 40).

Kada želimo da izvršimo skok na neku naredbu, koristićemo naredbu GO TO iza koje ćemo navesti broj naredbe (linije) na koju želimo da izvršimo skok. Primetimo da ovako kako je napravljen program njegov deo (II) bi trebalo da se izvršava beskonačan broj puta. To je stoga što smo formirali petlju, ali i nismo postavili nikakav uslov za izlaz iz nje, što se vidi na sl. 3.1.



Sl. 3.1 — Dijagram programa "Obračun kamate po stopi 7,5%"

Program je posle dužeg vremena ipak završio sa radom. Razlog završetka je to što je sadržaj promenljive (IZNOS) stalno uvećavan sve do momenta kada je vrednost koja treba da se upiše premašila kapacitet ćelije u koju se vrši upis. O tome nam i svedoči sledeća poruka o greški :

NUMBER TO BIG, 50 : 1

Primetimo još jednu važnu stvar. I u pisanju programa (1.075) i pri izlazu rezultata (.) se koristi kao simbol za razdvajanje celih i decimala, a ne (,) kao što je kod nas uobičajeno.

Izmenimo postojeći program tako što ćemo pri prikazu iznosa uvećanog za kamatu dodati i prikaz godine na koju se taj iznos odnosi :

```
10 REM OBRACUN KAMATE PO STOPI 7,5%
20 PRINT "OBRACUN KAMATE PO STOPI 7,5%"
30 LET IZNOS = 1823
35 LET GODINA = 1
40 PRINT GODINA ; " " ; IZNOS
50 LET IZNOS = IZNOS + IZNOS * 0.075
55 LET GODINA = GODINA + 1
60 GO TO 40
```

	godina	iznos
POČETNA VREDNOST NAREDBE 30, 35	1	1823
PRVO IZVRŠENJE NAREDBI 50, 55	2 = 1+1	1959,725 =1823 * 1,075
DRUGO IZVRŠENJE NAREDBI 50, 55	3 = 2+1	2106,704 = 1959,725 * 1,075
TREĆE IZVRŠENJE NAREDBI 50, 55	4 = 3+1	2264,707 = 2106,704 * 1,075

Sl. 3.2 — Promena sadržaja promenljivih GODINA i IZNOS u programu "Obračun kamate po stopi 7,5%"

Verovatno imate neku ušteđevinu. Izmenite postojeći program da biste videli kako se kreće kamata. Takođe kombinujte malo sa različitim vrednostima kamatne stope.

Već smo rekli da SPECTRUM koristi aritmetiku pokretnog zareza i da radi sa 9 značajnih cifara.

To u nekim slučajevima morate imati na umu da biste dobili rezultat koji je tačan. Unesite naredbu :

PRINT 1E20 + 1 — 1E20

i kao rezultat dobićemo 0.

Uradimo sada PRINT 1E20 — 1E20 + 1.

Rezultat je sada 1.

Ovaj na izgled nelogičan rezultat proističe iz toga što se zapis vrši na ograničenoj dužini, i to od 9 cifara. Zato izraz 1E20+1 daje kao rezultat 1E20.

```
  1E20 = 1·10^20 = |1|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|
+    1 = 1·10^0 +  |0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|1|
                 = |1|0|0|0|0|0|0|0|0|
                                     |0|0|0|0|0|0|0|0|0|0|0|0|
         1·10^20 = |1|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|
```

Sl. 3.3 — Rezultat operacije sabiranja 1E20 + 1 je 1E20

Nenumeričke ili STRING promenljive mogu sadržati bilo koji znak: brojeve, slova, simbole, blanko itd.

Naziv nenumeričke promenljive može biti isključivo sastavljen od jednog slova koje sledi simbol $. Na primer A$, C$, M$, . . . Ako želimo da STRING promenljivoj dodelimo neki sadržaj, možemo takođe koristiti naredbu LET, kao što ćemo to uraditi u programu koji sledi:

```
10 REM NENUMERICKE PROMENLJIVE
20 LET A$ = "KVADRATNI KOREN OD BROJA  "
30 LET B$ = "JE  "
40 LET BROJ = 50
50 PRINT A$ ; BROJ ; B$ ; SQR(BROJ)
```

Prepravite ovaj program tako da se štampaju svi brojevi od 100 na dalje sa korakom 100 i njihovi kvadratni koreni. Izvršenje programa prekinite pritiskom na dirku BREAK.

U okviru naredbe LET i u slučaju kada je u pitanju dodeljivanje nekog sadržaja STRING-promenljivoj, moguće je naći znak (+). Na primer :

```
20 LET A$ = "MEDJU"
25 PRINT A$
30 LET B$ = "SPRAT"
35 PRINT B$
40 LET C$ = A$ + B$
45 PRINT C$
```


U prvom redu izlazi MEDJU, sledećem SPRAT, a u trećem MEDJUSPRAT. Ako se kod naredbe LET koja se odnosi na STRING-promenljive pojavi znak (+), sadržaj prijemne promenljive biće sastavljen iz sadržaja prve promenljive iza koga neposredno sledi sadržaj druge.

naredba	posle izvršenja naredbe
20 LET A$ = "MEDJU"	A$ MEDJU
30 LET B$ = "SPRAT"	B$ SPRAT
40 LET C$ = A$ + B$	C$ MEDJUSPRAT

Sl. 3.4 — "Sabiranje" dve nenumeričke promenljive

3.2. NAREDBA INPUT

U svim do sada urađenim programima posebne vrednosti za promenljive bile su zadate u programima. Tako, na primer, u programu koji računa kamate zadali smo naredbom LET početnu vrednost za IZNOS (tj. za visinu štednog uloga). Ako smo želeli da izračunamo kamatu za neki drugi štedni ulog, morali bismo da menjamo program. To nam ne bi bio u ovom slučaju poseban problem, ali to svakako nije uvek slučaj.

Druga, ali važnija primedba je to što program treba da bude nezavisan od podataka. Mora postojati mogućnost da promenljiva dobije željeni sadržaj bez promena programa. Za te operacije se koristi naredba INPUT (ulaz) koja ima funkciju ulaza podataka za vreme izvršenja programa. Podaci se unose posredstvom tastature računara. Kada se za vreme izvršenja programa naiđe na naredbu INPUT, izvršenje se zaustavlja, sačeka da se podaci unesu, posle čega se nastavlja sa izvršenjem programa.

Program koji sledi zahteva sprovođenje unosa dva podatka, tj. dva sabirka. Program je tako napravljen da zahteva unošenje prvog sabirka, posle toga se pritiska ENTER, zatim se traži

drugi broj koji se unese, a potom se pritisne ENTER. Program ispisuje sabirke i rezultat :

```
10 REM NAREDBA INPUT
20 INPUT X
30 PRINT, X
40 INPUT Y
50 PRINT, Y
60 LET Z=X+Y
70 PRINT," ▬ "
80 PRINT, Z
```

Primetite da smo u naredbi 70 koristili grafički znak.

Veoma je korisno da se, pre nego što tražimo da se izvrši neka operacija ulaza, izvrši prikaz naziva, odnosno značenja promenljive za koju se vrši unošenje. To nam nije bilo baš neophodno u prethodnom slučaju, ali da smo u primeru imali program koji râdi oduzimanje bilo bi korisno reći da se prvo unosi vrednost za umanjenik, a potom za umanitelj. To ćemo uraditi u sledećem primeru. Kao što vidimo, funkciju prikaza takođe sprovodi naredba INPUT:

```
10 REM NAREDBA INPUT
20 INPUT "UNESITE VREDNOST PRVOG SABIRKA";X
30 PRINT, X
40 INPUT "UNESITE VREDNOST DRUGOG SABIRKA";Y
50 PRINT, Y
60 LET Z=X+Y
70 PRINT, " ▬ "
80 PRINT, Z
```

Izmenite program koji vrši obračun kamate tako da se pomoću naredbe INPUT preuzimaju posebne vrednosti za visinu štednog uloga i kamatnu stopu.

3.3. ARITMETIČKI OPERATORI I IZRAZI

Aritmetički operatori koje koristi BASIC SPECTRUM su +, —, *, /. (* se koristi za označavanje operacije množenja, a /za deljenje). Kombinujući, na dozvoljen način, promenljive i konstante pomoću operatora dobijamo aritmetičke izraze. Jedna od osnovnih osobina izraza je to da se može svesti na jednu vrednost. Tako, na primer, ako je izraz SUMA*5/2, svođenje na jednu vrednost vrši se na sledeći način: sadržaj promenljive SUMA pomnoži sa

5, a potom dobijenu vrednost deli sa 2. Dobijeni rezultat predstavlja vrednost aritmetičkog izraza.

Kod računara SPECTRUM postoji 16 nivoa prioriteta izvršavanja operacija. Operacije * i / imaju nivo prioriteta 8, a + i — prioritet 6. Ovi prioriteti su bitni pri svođenju aritmetičkog izraza na jednu vrednost. Izvršavaju se prvo operacije većeg nivoa prioriteta, i to sleva nadesno. Na primer:

A + B / C

tumači se kao: podeli sadržaj promenljive B sa sadržajem promenljive C i dobijeni rezultat saberi sa A, $\left(\text{tj. } a+\frac{b}{c}\right)$.

A*B—C*D/E/F

tumači se kao: pomnoži sadržaj promenljive A sa sadržajem promenljive B i to smatraj za međurezultat — 1 (MR1). Pomnoži sadržaj promenljive C sa sadržajem promenljive D (MR2). Podeli sadržaj međurezultata – 2 (MR2) sa E (MR3). Podeli sadržaj međurezultata —3 (MR3) sa sadržajem promenljive F (MR4). Oduzmi MR4 od MR1, tj. $\left(a\cdot b-\frac{c*d}{ef}\right)$.

Vidimo da ako su u pitanju operacije istog stepena prioriteta, one se izvršavaju sleva nadesno. Navedeni međurezultati se nigde eksplicitno ne pojavljuju, ali predstavljaju put kojim se ide u svođenju aritmetičkog izraza na jednu vrednost koju programer mora poznavati da bi mogao da izraz korektno napiše.

Pođimo sada obrnutim putem, tj. pokušajmo da izraz $\frac{a\cdot b}{c\cdot d}$ napišemo u formi aritmetičkog izraza u BASIC-u. Napišimo ga prvo kao :

A * B / C * D

što na izgled liči na odgovarajuću zadatu formu. Pokušajmo sada to da proverimo. Izraz se tumači kao: pomnoži sadržaj promenljive A sa sadržajem promenljive B (MR1). Podeli sadržaj međurezultata —1 sa sadržajem promenljive c (MR2). Pomnoži sadržaj međurezultata —2 sa sadržajem promenljive D, tj. $\left(\frac{a\cdot b\cdot d}{c}\right)$.

Vidimo da smo napravili grešku. Pravilno napisan izraz je sledeći:

A*B/C/D.

U okviru aritmetičkih izraza mogu se pojaviti i zagrade, i to mala otvorena "(" i mala zatvorena zagrada ")". Ako se u izrazu pojave zagrade, prvo se izvršavaju operacije u zagradama.

(A + B) / C

tumači se kao: saberi sadržaj promenljive A sa sadržajem promenljive B i dobijeni rezultat podeli sa sadržajem promenljive C $\left(\text{tj. } \frac{a+b}{c}\right)$.

Ako u izrazu ima nekoliko zagrada, prvo se izvršavaju operacije u najobuhvaćenijem pâru zagrada, odnosno sleva nadesno.

(A*B)*(3—(A+B)/6)

tumači se kao: prvo saberi sadržaj promenljivih A i B (MR1). Pomnoži A i B (MR2) – prva zagrada. Podeli MR1 sa 6 (MR3) — prioritet deljenja nad oduzimanjem. Oduzmi MR3 od 3 (MR4). Pomnoži MR2 i MR4, tj. $a \cdot b \cdot \left(3 - \frac{A+B}{6}\right)$.

3.4. *MATEMATIČKE FUNKCIJE*

U BASIC-u postoje i osnovne matematičke funkcije, i to:

stepenovanje ↑

EXP	e^x
LN	ln (prirodni logaritam)
SIN	sin (sinus)
COS	cos (kosinus)
TAN	tg (tangens)

Takođe postoje i inverzne trigonometrijske funkcije:

ASN	arcsin (arkus sinus)
ACS	arccos (arkus kosinus)
ATN	arctg (arkus tangens)

Kada se u okviru nekog aritmetičkog izraza pojavi i matematička funkcija, onda izračunavanje matematičke funkcije ima prvenstvo nad operacijama *, /.

Matematičke funkcije se koriste tako što se navede naziv funkcije i iza toga se napiše vrednost argumenta. Argument može ta-

kođe da bude složen aritmetički izraz, i tada se piše u zagradi. Pogledajmo nekoliko primera:

2↑1 = 2
2↑2 = 2·2
2↑3 = 2·2·2
2↑4 = 2·2·2·2

Prema tome, a^b pisaćemo kao A↑B. Podsetimo se da je kod stepenovanja:

a↑0 = 1
a↑(—1) = 1/a
a↑(—b) = 1/a↑b
a↑(1/b) = $\sqrt[b]{a}$

Funkciju kvadratnog korena možemo pisati na dva načina:

SQR a   ili   a↑(1/2)

Uradimo sada jedan zadatak. Treba napisati program koji kao rezultat štampa X1 i X2 rešenja kvadratne jednačine $ax^2+bx+c=0$. Vrednosti za a, b, c se preuzimaju naredbom INPUT.

```
10 REM KVADRATNA JEDNACINA
20 INPUT "UNESITE VREDNOST ZA A";A
30 INPUT "UNESITE VREDNOST ZA B";B
40 INPUT „UNESITE VREDNOST ZA C";C
50 LET X1=(—B+SQR(B↑2—4*A*C))/(2*A)
60 LET X2=(—B+SQR(B↑2—4*A*C))/(2*A)
70 PRINT "X1=";X1;"X2=";X2
80 GO TO 20
```

Program sadrži liniju 80 (naredbu GO TO), te može da unos vrednosti, računanje i prikaz izvrši više puta. Program prekinite pritiskom na dirku BREAK. Obratite pažnju na to da ovaj program pretpostavlja takav skup a, b, c na ulazu koji daje $b^2—4ac \geqslant 0$, tj. pozitivnu diskriminantu. Kakva je zaista diskriminanta ovde ne ispitujemo, zato što još uvek ne znamo kako da postavimo pitanje. Zato se može desiti da na ulazu zadamo takav skup vrednosti

za a, b, c, koji vodi do negativne potkorene veličine. U tom slučaju javlja se sledeći komentar o greški:

INVALID ARGUMENT

Trigonometrijske funkcije sin, cos i tg imaju argument koji se izražava u radijanima. Znamo da pun krug ima 2Pi radijana, tj. 360°. Prema tome možemo uvek, ako nam je zadat ugao u stepenima, naći odgovarajući u radijanima:

$$1^\circ = \frac{Pi}{180}$$

Spectrum raspolaže takođe i funkcijom Pi=3.14... (Pokušajte da date PRINT PI).

Napišimo sada program koji će da pretvori stepene u radijane. Pretpostavlja se da se naredbom INPUT preuzima broj stepeni, i to u celim brojevima.

```
10 REM * STEPEN / RADIJAN ***
20 INPUT "UNESITE BROJ STEPENI"; STEPEN
30 LET RADIJAN = PI / 180 * STEPEN
40 PRINT STEPEN; "STEPENI JE"; RADIJAN;
   "RADIJANA"
50 GO TO 20
```

Program prekinite naredbom BREAK. Pokušajte da prepravite program tako da se broj radijana izražava prema broju Pi.

Poznato nam je iz matematike da je:

$$\mathrm{tg}\,\alpha = \frac{2\,\mathrm{tg}\frac{\alpha}{2}}{1 - \mathrm{tg}^2\frac{\alpha}{2}}$$

$$\alpha = \mathrm{arc\,tg}\left(\frac{2\,\mathrm{tg}\frac{\alpha}{2}}{1 - \mathrm{tg}^2\frac{\alpha}{2}}\right)$$

Napravimo program koji će da učitava vrednosti za α u radijanima, zatim po navedenoj formuli ponovo izračunava α i prikazuje izračunatu vrednost. Na ovaj način možemo da vidimo sa kolikom se preciznošću računaju trigonometrijske funkcije.


```
10 REM ** ALFA I ALFA ***
20 INPUT "UNETI VREDNOST ZA ALFA U RAD" 'ALFA
30 LET ALFA1 =
   ATN(2*TAN(ALFA/2)/(1-TAN(ALFA/2)↑2))
40 PRINT "ALFA="; ALFA; "ALFA1="; ALFA1;
   "RAZLIKA="; ALFA-ALFA1
50 GO TO 20
```

Program prekinite naredbom BREAK. Posmatrajte i komentarišite kako se menja RAZLIKA zavisno od ALFA.

Izmenite program tako da se unos i prikaz vrši u stepenima, a ne u radijanima. Kako se ponaša program u okolini 90 stepeni?

Funkcija LN je prirodni logaritam, tj. LN X = $\log_e$ X. Ako nam je potrebna vrednost logaritma za neku drugu osnovu, onda ga moramo izvesti na osnovu onoga što znamo iz matematike — $\log_a$ X = ln X/ln a.

3.5. FUNKCIJE INT I RND

Funkcija INT izračunava celobrojni deo argumenta. Ako se sprovede naredba PRINT INT(12.45), na ekranu će se kao rezultat operacije dobiti 12.

Funkcija INT ne vrši zaokruživanje argumenta. Tako će se kao rezultat naredbe PRINT INT(25.99) dobiti 25, a ne 26, kao što ste možda očekivali. Ako želimo da sprovedemo zaokruživanje, možemo ga programirati dodavanjem 0,5 na vrednost argumenta:

```
10 REM ZAOKRUZIVANJE
20 INPUT "UNESITE IZNOS"; IZNOS
30 PRINT "PO ZAOKRUZENJU"; INT (IZNOS+.5)
40 GO TO 20
```

Utvrdite sami kako funkcija INT vrši zaokruživanje negativnih brojeva.

Funkcija RND se veoma često koristi, naročito u programima igara. Ona generiše slučajne brojeve čija se vrednost kreće između 0 i 1, što se može videti ako se izvrši sledeći program:

```
10 PRINT RND
20 GO TO 10
```

Funkcija RND se može modifikovati, a tako se najčešće i koristi, da daje cele brojeve u nekom intervalu. Ako bismo želeli da se dobijaju celi brojevi u intervalu 1—10, uradićemo:

```
10 PRINT INT (RND*10)+1
20 GO TO 10
```

Izraz INT(RND*10) generiše slučajne brojeve 0—9. Posle dodavanja 1 opseg će biti 1 — 10.

LOTO je igra na sreću u kojoj se na slučajan način izvlači 5 od 36 brojeva (1 — 36). Sličniu igru možete odigrati i uz pomoć vašeg računara :

```
10 REM LOTO
20 FOR I=1 TO 5
30 PRINT "BROJ" ; I ; " JE " ; INT (RND * 36) + 1
40 NEXT I
```

Kada program završi sa radom, dobićete na ekranu pet LOTO brojeva u redosledu kako su izvučeni. Naredbe 20 i 40 koje ćemo obrađivati kasnije omogućavaju da se naredba 30 obavi pet puta.

3.6. *VEŽBE*

1. Napisati program koji računa vrednosti izraza :

a) $y=\dfrac{b-b^3-4\,ac}{2\,a^2}$

b) $y=(a-b^2)+\sin^2 x$

c) $y=\dfrac{(a-b)^2}{a^2+b^3-c}$

Posebne vrednosti za a, b, c i x preuzimaju se naredbom INPUT. Vrednost za x data je u stepenima.

2. Napisati programe koji računaju:

a) obim, površinu i dijagonalu pravougaonika čije su stranice a i b;

b) zadata je stranica jednakostraničnog trougla a — izračunati poluprečnik upisanog i opisanog kruga i površinu trougla;

c) poznate su stranice trougla a, b i c — izračunati površinu trougla i poluprečnik opisanog kruga;

Posebne vrednosti za a, b i c se preuzimaju naredbom INPUT.

3. Potrošač diže kredit na 100.000 din. uz 25% kamate. Otplata traje 24 meseca. Kolika će biti visina mesečne rate i koliki će biti ukupan dug ako je učešće 40%.

4. Napisati program koji računa aritmetičku i geometrijsku sredinu pet brojeva.

5. Napisati program koji računa rastojanje dve tačke A i B. Poznate su koordinate tačaka X_1, Y_1 za A i X_2, Y_2 za B.

6. Napišite program koji čita vrednosti za R, L, C i f računa i štampa Z i Φ.

$$Z=\sqrt{R^2+\left(\omega L-\frac{1}{\omega C}\right)^2}$$

$$\Phi=\mathrm{tg}^{-1}\frac{\omega L-\frac{1}{\omega C}}{R}$$

gde je $\omega = 2\pi f$.



Ispitivanje uslova

4.1. IF ... THEN

Do sada smo napisali nekoliko programa u kojima se neki broj naredbi pri izvođenju ponavlja "veći" broj puta. Taj "veći" broj bio je sa tačke gledišta samoga programa neodređen, odnosno (da za to postoje uslovi) te naredbe bi se izvršavale beskonačan broj puta. Programe smo pri izvršenju prekidali komandom BREAK. Ovo "spoljno" ili "nasilno" prekidanje se u praksi gotovo nikada ne koristi, a i mi smo to upotrebljavali samo zato što nismo znali kako to da uradimo na drugi način. U primerima koje smo do sada obrađivali to nije uticalo na ostvarenje funkcije programa.

Uzmimo, međutim, primer programa koji treba da računa prosečnu ocenu iz matematike učenika neke škole. Struktura tog programa je sledeća:

(I) Učitati ocenu pojedinog učenika i sabrati tu ocenu u promenljivu čiji sadržaj treba da predstavlja zbir ocena svih učenika; dodati jedinicu u promenljivu čiji sadržaj treba da predstavlja broj učenika za koje se prosečna ocena računa.

Očigledno je da postupak (I) treba da ponavljamo onoliko puta koliko ima učenika. Kada se unesu podaci za poslednjeg učenika (tj. njegova ocena iz matematike), Spectrum traži podatke za sledećeg. Pošto ga više nema, po analogiji sa dosadašnjim primerima pritiskamo BREAK. Šta se desilo? Program je završio sa radom i nije nam dao nikakav rezultat. To smo mogli i pretpostaviti jer:

$$\text{prosečna ocena} = \frac{\text{zbir svih ocena}}{\text{broj učenika}}$$

može da se izračuna tek kada se unesu podaci za sve učenike deljenjem dve već opisane promenljive, a mi smo prekinuli program pre nego što smo deljenje sproveli. Vidimo da se program mora podeliti u dva dela. Deo (I) koji se ponavlja onoliko puta koliko je učenika i deo (II) gde se računa prosečna ocena za navedeni način i prikazuje se na ekranu. Deo (II) programa treba da se izvrši kada se unesu ocene za sve učenike. Moramo, znači, imati naredbu koja će računaru, zavisno od nekog uslova (u ovom slučaju — da li su unesene sve ocene), omogućiti da izvrši izbor skupa naredbi koji treba da izvršava. To se postiže naredbom IF. Opšti oblik naredbe IF je:

IF uslov THEN naredba

i ima sledeće značenje: AKO (IF) je ispunjen uslov, TADA (THEN) radi naredbu. Ako uslov nije ispunjen, tada se izvršava naredba koja sledi naredbu IF:

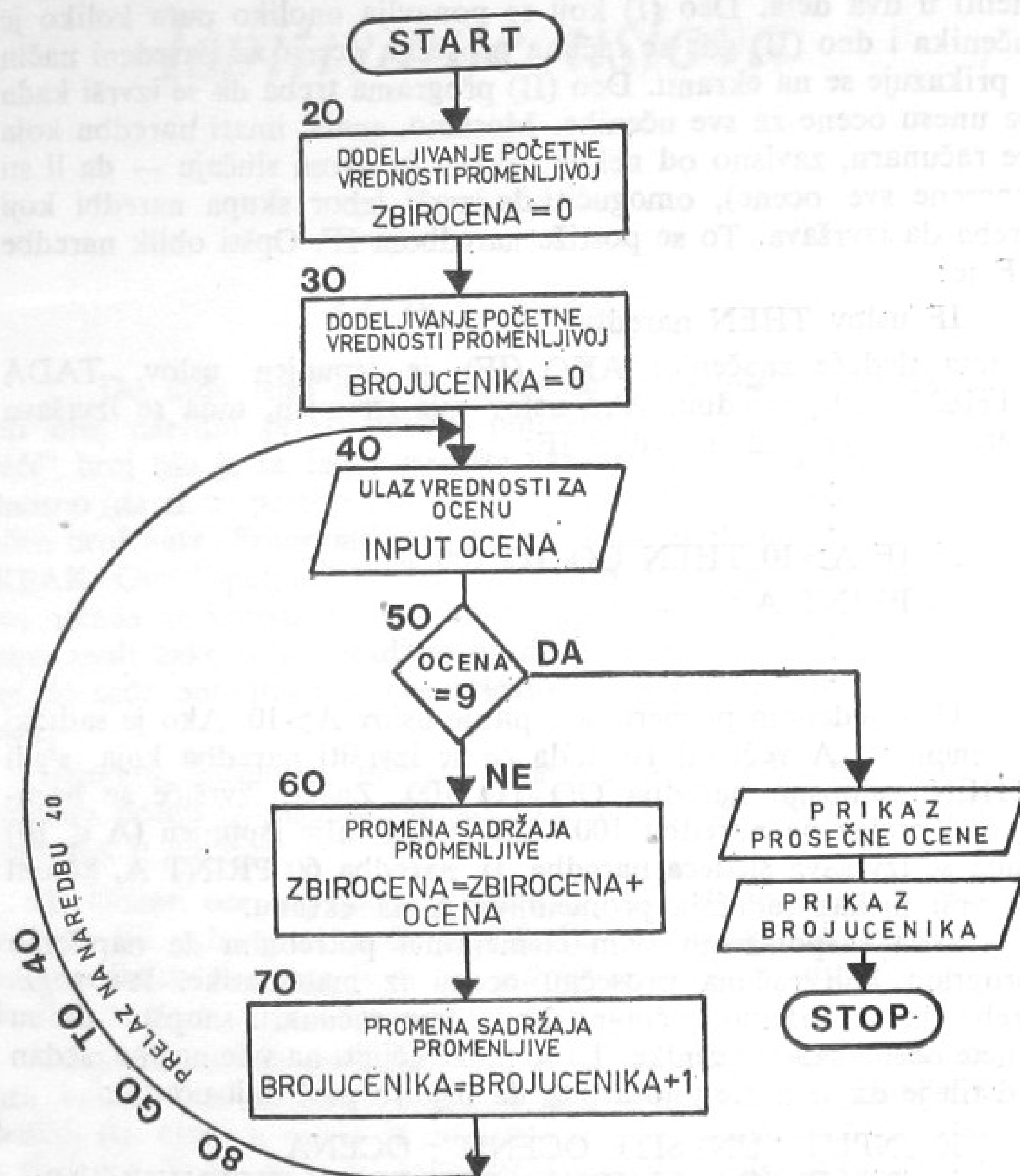
```
:
50 IF A>10 THEN GO TO 100
60 PRINT A ;
:
```

U navedenom primeru se ispituje uslov A>10. Ako je sadržaj promenljive A veći od 10, tada će se izvršiti naredba koja sledi THEN, odnosno naredba GO TO 100. Znači, izvršiće se bezuslovni prelaz na naredbu 100. Ako uslov nije ispunjen ($A \leq 10$) tada se izvršava sledeća naredba, tj. naredba 60 PRINT A, kojom se vrši prikaz sadržaja promenljive A na ekranu.

Sada raspolažemo svim elementima potrebnim da napišemo program koji računa prosečnu ocenu iz matematike. Pre toga treba da definišemo način na koji ćemo računaru saopštiti da su unete ocene za sve učenike. To se može učiniti na više načina. Jedan od tih je da se posle unosa pita da li je to poslednji učenik:

```
30 INPUT "UNESITE OCENU"; OCENA
40 INPUT "DA LI JE TO POSLEDNJI UCENIK"; A$
50 IF A$<> "DA" THEN GO TO 100
```

Loša strana ovoga rešenja je to što uvek posle svakog unosa moramo da odgovorimo na pitanje "DA LI JE TO POSLEDNJI UCENIK". Tako osoba koja unosi podatke o ocenama više vremena provede odgovarajući da to nije poslednji učenik, nego što unosi ocene. Druga mogućnost je da se, pošto se unese i obradi ocena za poslednjeg učenika, ponovo operacijom ulaza unese umesto vrednosti za ocenu neka karakteristična vrednost koja ima značenje kraja unosa. Ako ocene mogu da imaju vrednost 1—5, neka oznaka kraja bude 9. Tako dolazimo do sledećeg programa:



Sl. 4.1 — Dijagram programa "Prosečna ocena"


```
10 REM PROSECNA OCENA
20 LET ZBIROCENA = 0
30 LET BROJUCENIKA = 0
40 INPUT "UNESITE OCENU"; OCENA
50 IF OCENA = 9 THEN GO TO 90
60 LET ZBIROCENA=ZBIROCENA + OCENA
70 LET BROJUCENIKA = BROJUCENIKA + 1
80 GO TO 40
90 PRINT "PROSECNA OCENA JE"; ZBIROCENA/
   BROJUCENIKA
100 PRINT "BROJ UCENIKA JE" ; BROJ UCENIKA
```

	ZBIROCENA	BROJUCENIKA	ULAZ ocena
DODELJIVANJE POČETNE VREDNOSTI NAREDBE 20,30	0	0	
PRVO IZVRŠAVANJE NAREDBI 40,60,70	4 (=0+4)	1 (=0+1)	4
DRUGO IZVRŠAVANJE NAREDBI 40,60,70	9 (=4+5)	2 (=1+1)	5
TREĆE IZVRŠAVANJE NAREDBI 40,60,70	11 (=9+2)	3 (=2+1)	2
ČETVRTO IZVRŠAVANJE NAREDBE 40 PROUZROKUJE RAČUNANJE I PRIKAZ PROSEKA I STOP	PROSEK 11/3 = 3.66		9

Sl. 4.2 — Promena sadržaja promenljivih ZBIROCENA i BROJUCENIKA u programu "Prosečna ocena"

Naredba koja sledi naredbu IF ne mora biti naredba GO TO, već može biti i neka druga, kao u sledećem programu:

```
10 REM IF PRIMER
20 INPUT "UNESITE BROJ OD 1 DO 3" ; BROJ
30 IF BROJ = 1 THEN PRINT "JEDAN"
40 IF BROJ = 2 THEN PRINT "DVA"
50 IF BROJ = 3 THEN PRINT "TRI"
```

Operatori koji dozvoljavaju da se obavlja poređenje u okviru uslova su :

- = jednak
- > veći od
- < manji od
- <> nejednak
- >= veći ili jednak
- <= manji ili jednak

Spectrum dozvoljava i ispitivanje tzv. složenih uslova. Složeni uslov se formira dovođenjem u vezu prostih uslova preko logičkih operatora AND (i), OR (ili) i NOT (ne).

U prethodno napisanom programu i u programu koji računa prosečnu ocenu nije rešen problem unosa vrednosti za promenljivu BROJ koje ne spadaju u skup željenih vrednosti 1, 2, 3. Šta će se desiti ako startujete navedeni program i unesete vrednost 4. Prema tome kako je program napravljen neće se desiti ništa zato što će tome kako je program napravljen neće se desiti ništa zato što će odgovor na sva tri pitanja biti negativan. U najvećem broju slučajeva, u praksi, to nas ne zadovoljava, pa odmah posle unosa neke vrednosti želimo da utvrdimo da li je uneta vrednost korektna. To se u našem slučaju svodi na ispitivanje da li je sadržaj BROJ 1, 2 ili 3. Na osnovu toga formiraju se sledeći složeni uslovi:

BROJ = 1 OR BROJ = 2 OR BROJ = 3

BROJ je jednak 1 ili 2 ili 3. Ako je ovaj uslov ispunjen, uneta vrednost je korektna i tada treba ići na naredbu 30. Ako to nije slučaj, daćemo komentar o grešci i ići na naredbu 20. Promenite program tako što ćete uneti sledeće linije :

```
22 IF BROJ=1 OR BROJ=2 OR BROJ=3 GO TO 30
24 PRINT "UNETA VREDNOST NE VALJA " ; BROJ
26 GO TO 20
```

Primetite da će, kad izvršavate ovaj program, računar reagovati na svaku unetu nekorektnu vrednost tražeći da unesete vrednost koja odgovara uslovu zadatka.

Predpostavimo slučaj da želimo utvrditi da li je neka osoba stekla uslov za odlazak u prevremenu penziju. Da bi ostvarila to pravo, potrebno je da ima više od 60 godina starosti i više (ili jednako) od 10 godina radnog staža:


```
10 REM PENZIJA
20 INPUT "UNESITE VREDNOST ZA STAROST";
   STAROST
30 INPUT "UNESITE VREDNOST ZA RADNI STAZ" ;
   STAZ
40 IF STAROST>60 AND STAZ>=10 THEN GO TO 70
50 PRINT "NEMA USLOVA ZA PENZIJU"
60 STOP
70 PRINT "POSTOJE USLOVI ZA PENZIJU"
```

U naredbi 40 imamo složeni uslov koji se tumači kao starost veća od 60 i (AND) staž veći ili jednak 10. Uveli smo i jednu novu naredbu — naredbu STOP. Kada se pri izvršenju programa naiđe na STOP, program prestaje sa radom i šalje poruku koja ukazuje na broj linije na kojoj se nalazi naredba STOP koja je uslovila završenje programa. Vidimo da se program regularno završava na dva načina: (I) nailaskom na naredbu STOP (II) kada se dođe do fizičkog kraja programa (do poslednje naredbe u programu ako ona ne uključuje naredbu GO TO).

Operator NOT se koristi u slučaju kada želimo da izvršimo izmenu značenja nekog uslova. Pretpostavite da se možete videti sa vašim prijateljima svakog dana osim subote i nedelje — kada idete na izlet. Da bi se utvrdilo da li ste nekog dana slobodni, treba uneti redni broj dana u nedelji (ponedeljak = 1, utorak = 2,..., nedelja = 7):

```
10 REM SASTANAK
20 INPUT "UNESI REDNI BROJ DANA"; DAN
30 IF NOT (DAN=6 OR DAN=7) THEN GO TO 70
40 PRINT "ZAO MI JE ALI ZA"
50 PRINT "VIKEND IDEM NA IZLET"
60 STOP
70 PRINT "VAZI VIDECEMO SE"
```

Pokušajte da prepravite prethodne programe tako što ćete tamo gde je to moguće promeniti logičke operatore. Takođe razbijte složene uslove u odgovarajući broj prostih uslova.

Složeni uslovni izraz može da sadrži zajedno AND i OR operatore i oni se tada zovu kombinovani :

STAROST=10 AND RAZRED=3 OR OCENA=5

Isto kao što postoji prioritet među aritmetičkim operatorima, postoji i prioritet izvršenja operatora AND nad operatorom OR. Prema tome, gornji izraz se tumači kao tačan ako je zadovoljen

složeni uslov da je starost jednaka 10 godina i da je razred jednak 3 ili ako je zadovoljen prost uslovni izraz da je ocena jednaka 5.

U okviru uslova koji se ispituje može se pojaviti i aritmetički izraz, na primer :

```
30 IF A=B+C THEN GO TO 200
```

sledećeg značenja: ako je sadržaj promenljive A jednak zbiru sadržaja promenljivih B i C idi na naredbu 200.

Mi smo radili program koji računa rešenja kvadratne jednačine. Rekli smo da ulazne vrednosti moraju biti takve da je diskriminanta ≥0. Sada, kada znamo naredbu IF, takvo ograničenje možemo postaviti u sâm program. Posle ulaza vrednosti za A, B i C ispitajte da li je diskriminanta ≥0, pa ako jeste izračunajte vrednosti za X1 i X2. Ako je diskriminanta negativna, pošaljite odgovarajuću poruku na ekran (PRINT) i zaustavite rad izvršenja programa (STOP).

Pri ispitivanju uslova SPECTRUM nas ne ograničava samo na relacije numeričkog tipa. Možemo pitati:

```
60 IF A$="DA" THEN GO TO 260
```

odnosno porediti sadržaj nenumeričke promenljive A$ sa nenumeričkom konstantom DA. Poređenje toga tipa se koristi i u sledećem primeru u kome ispitivanjem sadržaja promenljive A$ želimo da se obezbedimo od mogućnosti da nam neko neovlašćeno izvršava program. Vidimo: ako nije uneto naredbom INPUT naše ime (u opštem slučaju lozinka bilo koje vrste), izvršava se naredba NEW čije dejstvo već poznajemo:

```
5 INPUT A
6 IF A$<>"PERA PERIC" THEN NEW
```

Ove dve naredbe unesite u bilo koji program. Prvi put unesite PERA PERIC. Program će se izvršiti na uobičajeni način. Startujemo ponovo isti program. Umesto PERA PERIC unesimo MARKO MARKOVIĆ. Ovoga puta aktiviraće se naredba NEW i program će se sam uništiti. To je posledica izvršenja naredbe NEW, odnosno ispunjenja prethodno postavljenog uslova.

Završimo ovo poglavlje izmenom programa koji vrši računanje korena kvadratne jednačine. Ovaj program ste već izmenili time što ste u njega uneli programsku kontrolu znaka diskriminante. Promenimo sada program utoliko da može da računa vrednosti rešenja za nepoznat broj trojki, A, B, C, pri čemu se zna da, kada

želimo da prekinemo postupak, unesemo za A, B i C vrednost 0. Program je sledeći:

```
10 REM KVADRATNA JEDNACINA
20 INPUT "UNESITE VREDNOST ZA A"; A
30 INPUT "UNESITE VREDNOST ZA B"; B
40 INPUT "UNESITE VREDNOST ZA C"; C
50 IF A=0 AND B=0 AND C=0 THEN STOP
60 IF A<>0 THEN GO TO 90
70 PRINT "JEDNACINA NIJE KVADRATNA"
80 PRINT "RESENJE JE X=";–C/B
85 GO TO 20
90 LET D=B↑2–4*A*C
100 IF D=0 THEN GO TO 140
110 IF D>0 THEN GO TO 170
120 PRINT "REALNA RESENJA NE POSTOJE"
130 GO TO 20
140 PRINT "JEDNACINA IMA JEDNO RESENJE"
150 PRINT "RESENJE JE X=";–B/(2*A)
160 GO TO 20
170 LET X1=(–B+SQR(D))/(2*A)
180 LET X2  =(–B–SQR(D))/(2*A)
190 PRINT "JEDNACINA IMA DVA RESENJA"
200 PRINT "RESENJA SU X1=";X1;"X2=";X2
210 GO TO 20
```

4.2. INTERAKTIVAN RAD

Najveći broj programa sa kojima ste se do sada sretali, pogotovu programi igara, predstavljaju "interaktivne programe". Njihova karakteristika je stalna interakcija između računara, tj. programa koji se izvodi i vas koji se u tom momentu služite računarom. Programi toga tipa u mnogim svojim tačkama zahtevaju od vas neki odgovor da bi se dalja akcija odvijala u nekom od mogućih pravaca.

Najprostiji slučaj takve interakcije su programi u kojima igrač ima mogućnost da pritiskanjem na određene dirke na tastaturi pomera jednu od figura sa ekrana i na taj način izbegava razne prepreke, opasnosti itd. To su igre tipa "Horacije na skijanju",

gde igrač ima mogućnost da pritiskom na jednu ili drugu dirku vrši pomeranje figure Horacija ulevo odnosno udesno. Na taj način se omogućava Horaciju da pravilno prolazi kroz kapije postavljene na skijaškoj stazi i da tako izbegava drveće ili hupsere koji se na njoj nađu. U program su između ostalog ugrađena i dva pitanja: da li želite da Horacije ide levo, da li želite da Horacije ide desno? Ta pitanja nam se eksplicitno ne pojavljuju na ekranu zato što bi usporavale rad programa i zato što je ovde dijalog računar—igrač jako prost i svodi se na pritiskanje dve dirke.

Dijalog sa računarom može biti i mnogo složeniji, u šta su se uverili svi koji su igrali igre tipa Hobit. Tamo se, pošto su igre mnogo složenije, vodi dijalog rečima kao:

WHAT NOW?
(računar: Šta sada?)
N
(Vi: Idem u pravcu severa N=NORD)
YOU CAN'T GO THAT WAY
(računar: Ne možete ići u tom pravcu)
JUMP IN STREAM
(Vi: skoči u potok)
YOU MUST BE JOKING. J CAN'T BREATHE UNDERWATER!
(računar: Vi se šalite. Ja ne mogu da dišem ispod vode)
WHAT NOW?
(računar: Šta sada?)
OPEN BOX
(Vi: Otvorite kapiju)
O.K. THE BOX IS OPEN. INSIDE YOU CAN SEE AN AQUALUNG.
(računar: Kutija je otvorena. Unutra možete videti opremu za ronjenje)
Itd.

Kod igara ove vrste dijalog igrač-računar je očigledan. Problem je kod ovih igara što se od igrača zahteva dobro poznavanje engleskog jezika, a veoma često i poznavanje same priče na kojoj je igra zasnovana.

Ako ste nekada kupovali kartu za avion na šalteru koji je opremljen računarskim terminalima primetili ste da se sličan dijalog vodi između službenika i računara.

Računar:	ZA KOJE MESTO I ZA KOJI DATUM ZELITE REZERVACIJU?
Službenik:	BEOGRAD 28. 04. 84.
Računar:	Za KOJI LET?
Službenik:	YU238
Računar:	ZALIM. POPUNJENO. ZELITE LI NEKI DRUGI LET?
Službenik:	NE

Možemo reći da se "interaktivno programiranje" bavi najviše formama dijaloga, načinom kako računaru dati neku informaciju, odnosno šta učiniti da računar tu informaciju korektno i brzo obradi i vrati natrag na ekran da bi dijalog bio nastavljen. Pri tome je važno da se informacije u oba smera jasno saopštavaju, tako da ne dolazi do "zabune" ni kod korisnika ni kod samog računara.

Pokažimo taj tip komunikacije na jednom veoma jednostavnom primeru.

```
120 INPUT "UNESITE JEDAN BROJ"; A
130 GO TO 120
```

Pustite program da radi. Pošto ste već uradili nekoliko programa, vi ćete unositi vrednosti 1,20,50 itd. dok vam sve to ne postane dosadno. Razmislite da li postoji mogućnost da neki vaš prijatelj koga računari ne interesuju kao odgovor na pitanje UNESITE JEDAN BROJ — kuca JEDAN (naravno slovima). Ovde će nesporazum biti obostran. Korisniku nije jasno da računar traži neku numeričku vrednost, a računar dobijenu informaciju ne može da koristi. Slično bi bilo kada bismo kao broj uneli 3,14 ili 1.000 umesto 1000. Iz ovih razloga u programiranju se uvek poklanja velika pažnja (i) jasnom saopštavanju onoga što se očekuje da korisnik unese i (ii) kontroli unetog sadržaja.

Tako će veliki broj nesporazuma otkloniti novi oblik naredbe 120:

```
120 INPUT"UNESITE JEDAN CEO BROJ OD 0 DO 9"; A
```

Neke vidove kontrole već smo do sada vršili, no da bismo mogli nastaviti upoznajmo nekoliko novih naredbi.

CODE funkcija se primenjuje na STRING-promenljive i daje kôd prvog znaka u string-promenljivoj. Svakom znaku odnosno naredbi odgovara neki kôd. Pregled kodova i odgovarajućih znakova dat je u prilogu 1. Naredba:

```
10 PRINT CODE "AB"
```

prikazaće na ekranu 65, što je kôd za A.

CHR$ funkcija je inverzna funkcija CODE. Njen argument je broj koji predstavlja vrednost kôda, a efekat znak koji odgovara kôdu. Naredba:

```
10 PRINT CHR$65
```

prikazaće na ekranu slovo A.

Funkcija CHR$ se često koristi za prikaz grafičkih znakova. Prikaz figure jahača sa slike 2.2 možemo dobiti na sledeći način:

```
10 PRINT CHR$128;CHR$128;CHR$143;CHR$128;CHR$128
20 PRINT CHR$140;CHR$136;CHR$133;CHR$128;CHR$128
30 PRINT CHR$128;CHR$138;CHR$133;CHR$128;CHR$128
40 PRINT CHR$128;CHR$139;CHR$135;CHR$135;CHR$128
50 PRINT CHR$128;CHR$138;CHR$128;CHR$133;CHR$128
```

LEN funkcija računa dužinu nenumeričke promenljive.

```
10 LET A$ = "BEOGRAD"
20 PRINT LEN A$
```

Rezultat ovih naredbi biće 7.

STR$ funkcija konvertuje numeričke u nenumeričke promenljive:

```
10 LET A = 234
20 PRINT STR$A
```

Dobićemo ponovo na ekranu 234. Razlika je u tome što je A numerička promenljiva zapisana u formi pokretnog zareza, a STR A$ nenumerička promenljiva čiji je sadržaj "234".

VAL funkcija je inverzna funkcija STR$. Kod nje se kao argument javlja nenumerička promenljiva čiji sadržaj može biti aritmetički izraz. Rezultat funkcije je odgovarajuća numerička vrednost:

```
VAL "3*3" = 9
VAL "5+6/3" = 7
```

Ranije smo već napisali INPUT naredbu 120 koja korisniku daje više podataka o tome kakav broj treba da unese. Dogradimo taj "program" dêlom koji se odnosi na kontrolu:

```
120 INPUT "UNESITE JEDAN CEO BROJ OD 0 DO 9";A$
130 IF NOT (CODE A$<48 OR CODE A$>57) THEN
    GO TO 170
140 PRINT "BROJ NE VALJA"; A$
```



```
150 PRINT "POKUSAJTE PONOVO"
160 GO TO 120
170 LET A=VAL A$
180 PRINT A
```

Primetimo da se ovoga puta uz INPUT naredbu pojavljuje nenumerička promenljiva A$. Kodovi cifara 0 – 9 su 48 do 57. Tako naredbom 130 vršimo kontrolu unetog broja. Ako je uneta vrednost korektna, korišćenjem VAL funkcije unesena vrednost postaje sadržaj numeričke promenljive A.

Problem ovoga programa je u tome što funkcija CODE daje kôd samo prvog znaka. To znači da će kontrola u liniji 130 reagovati samo kada se na prvoj poziciji nađe neki neželjen znak. Ako se, na primer, unese 2B84X, razmatraće se samo CODE od 2 i smatraće se da je unos u redu. Ono što moramo uneti u program je mogućnost da vršimo kontrolu svakog unetog znaka. To se može postići na sledeći način:

```
10 INPUT A$
20 LET N=1
30 IF N>LEN A$ THEN GO TO 70
40 IF CODE A$ (N TO N)<48
   OR CODE A$ (N TO N)>57
   THEN PRINT "BROJ NE VALJA": GO TO 10
50 LET N=N+1
60 GO TO 30
70 LET A=VAL A$
```

Ovaj program vrši kontrolu svakog od unetih znakova — da li predstavlja cifru 0 — 9. To je omogućeno opcijom TO korišćenom u naredbi 40 uz nenumeričku promenljivu A$.

A$(N TO N) ima značenje N-tog znaka promenljive A$.

Ako je A$="BEOGRAD", a N=1 A$(N TO N)=A$(1 TO 1)="B";N=3 A$(N TO N)=A$(3 TO 3) = "0". Takođe je A$(5 TO 6)="RA", A$(TO 3) = "BEO", A$(3 TO) = "OGRAD".

Naredbama 20, 30, 50 i 60 obezbedili smo da se naredba 40 obavi onoliko puta koliko znakova sadrži promenljiva A$(LEN A$). Pri tome, prvi put kada se naredba 40 izvršava N=1, drugi put N=2, treći put N=3 itd.

Pored naredbe INPUT korisnik može saopštavati informacije računaru i naredbom INKEY$. Naredbom INPUT može se uneti

više znakova, ali je neophodno pritisnuti dirku ENTER da bi se računaru saopštilo da je unos završen. Suprotno tome, sa INKEY$ se može uneti samo jedan znak, ali je povoljno to što se ne pritiska ENTER. U pomenutoj igri "Horacije na skijanju" nalozi za pokretanje figure Horacija se primaju uz pomoć INKEY$ zato što je kod igara ovoga tipa bitna brzina (ne pritiska se ENTER), a sa druge strane igrač saopštava računaru malo informacija (najčešće: levo ili desno, gore ili dole). U igrama u kojima je "razgovor" čovek-računar bogatiji i gde nije toliko bitna brzina kojom se nešto saopštava, informacije se primaju posredstvom naredbe INPUT.

INKEY$ je funkcija bez argumenata i čita stanje dirki sa tastature. Ako ste pritisnuli samo jednu dirku (ili dirku SHIFT i neku drugu dirku), tada je efekat INKEY$ funkcije znak koji odgovara dirki u L režimu rada. U drugom slučaju rezultat je prazan string. Prazan string ili prazna nenumerička promenljiva se označava sa " ".

Unesite sledeći program. Pri izvršavanju unesite brojeve 1—9 pritiskujući odgovarajuću dirku. Program će odgovoriti PRITISNULI STE 6, PRITISNULI STE 4, itd. Ako želite da završite rad programa, pritisnite 0:

```
10 REM INKEY$
20 PAUSE 100
30 LET A$ = INKEY$
40 PRINT "PRITISNULI STE"; A$
50 IF A$ = "0" THEN STOP
60 GO TO 20
```

Na sledeći način možemo reći računaru da čeka dok se ne pritisne neka dirka:

```
10 IF INKEY$ = " " THEN GO TO 10
```

U momentu kada se pritisne neka dirka program će preći na sledeću liniju zato što uslov neće biti zadovoljen.

Slično kontrolama ulaza koje smo obavljali uz naredbu INPUT, i u slučaju da se nalozi preuzimaju preko INKEY$ možemo odrediti opseg korektne vrednosti:

```
10 PAUSE 0
20 IF INKEY$ = CHR$ (13) OR
   INKEY$ = CHR$ (32) THEN
   GO TO 40
```



```
30 IF CODE (INKEY$)<65 OR
   (CODE (INKEY$)>90 AND
   CODE (INKEY$)<97) OR
   CODE (INKEY$)>122 THEN
   GO TO 10
40 PRINT INKEY$;
50 GO TO 10
```

Ako ovaj program pokrenete na izvršenje, videćete da se tada računar ponaša kao pisaća mašina. To je posledica proglašenja za ilegalne svih znakova osim A — Z (kôd 65—90) i a — z (kôd 97 — 122), odnosno velikih i malih slova.

Kôdu 13 odgovara ENTER, a 32 blanko. ENTER koristimo za prelaz u novi red.

4.3. VEŽBE

1. Napisati program koji prikazuje sve prirodne brojeve od 1 do 50, kao i njihov zbir.

2. Napisati program koji prikazuje tabelarno sve prirodne brojeve i njihove logaritme počev od broja 1 do zadanog N.

Napisati program koji prikazuje tabelarno sve uglove $-\pi/2$ do $\pi/2$ i njihove sinuse i kosinuse. Uzeti za korak 0.1. (Argument trigonometrijskih funkcija izražava se u radijanima).

4. Napisati program koji učitava realne brojeve i potom prikazuje samo pozitivne.

5. Za svakog učenika u razredu unose se dve vrednosti. Prvo se unese kôd "M" ili "Z", zavisno od toga da li je u pitanju dečak ili devojčica. Potom se unese podatak o visini dečaka, odnosno devojčice.

Prikazati na ekranu: broj dečaka, broj devojčice, prosečnu visinu dečaka i prosečnu visinu devojčice.

6. Napisati program koji prikazuje sumu niza:

$1+1/2+1/3+1/4 \cdots +1/n$

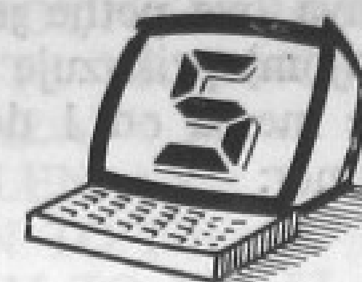
gde se vrednost za n preuzima naredbom INPUT.

7. Napisati program koji računa vreme u sekundama ako se vreme zada u danima, satima i sekundama. Operacije ulaza za dane, sate i sekunde, obračun i prikaz rezultata program treba da ponavlja

veći broj puta. Kraj rada programa postaviti kada je zadovoljen uslov DANI=0, SATI=0 i SEKUNDI=0.

8. Napisati program koji vrši prikaz rednog broja nekog dana u godini kada je poznat datum na koji se taj dan odnosi (6. februar je 37. dan u godini).

9. Ako su poznata dva datuma: datum—1 (godina, mesec, dan) i datum—2 (godina, mesec, dan), izračunati koliko između ta dva datuma ima dana. Datumi mogu biti u različitim godinama. Meseci imaju različit broj dana. Vrednosti za godine, mesece i dane preuzimaju se naredbom INPUT.



Programske petlje i rutine

5.1. NAREDBA FOR . . . NEXT

Par naredbi FOR . . . NEXT definiše programsku petlju u BASIC-u. Između ove dve naredbe (prva je FOR) nalazi se sekvenca instrukcija koja se može izvršiti veći broj puta — prema tome kako je to definisano naredbom FOR. NEXT naredba obeležava kraj navedene sekvence instrukcija. Naredba FOR ima sledeći oblik:

FOR upravljačka promenljiva = početna vrednost TO poslednja vrednost (limit).

Upravljačka promenljiva mora biti numerička, sa ograničenjem da joj naziv može biti samo jedno slovo. Na primer:

```
100 FOR J=1 TO 100
200 NEXT J
```

U ovom primeru J je upravljačka promenljiva koja uzima vrednosti 1, 2, 3, . . . 99, 100. Za svaku od tih vrednosti obavi se jednom sekvenca instrukcija koja se nalazi između 100—200. Kada sadržaj upravljačke promenljive premaši vrednost limita, prelazi se na naredbu koja sledi naredbu 200 NEXT J.

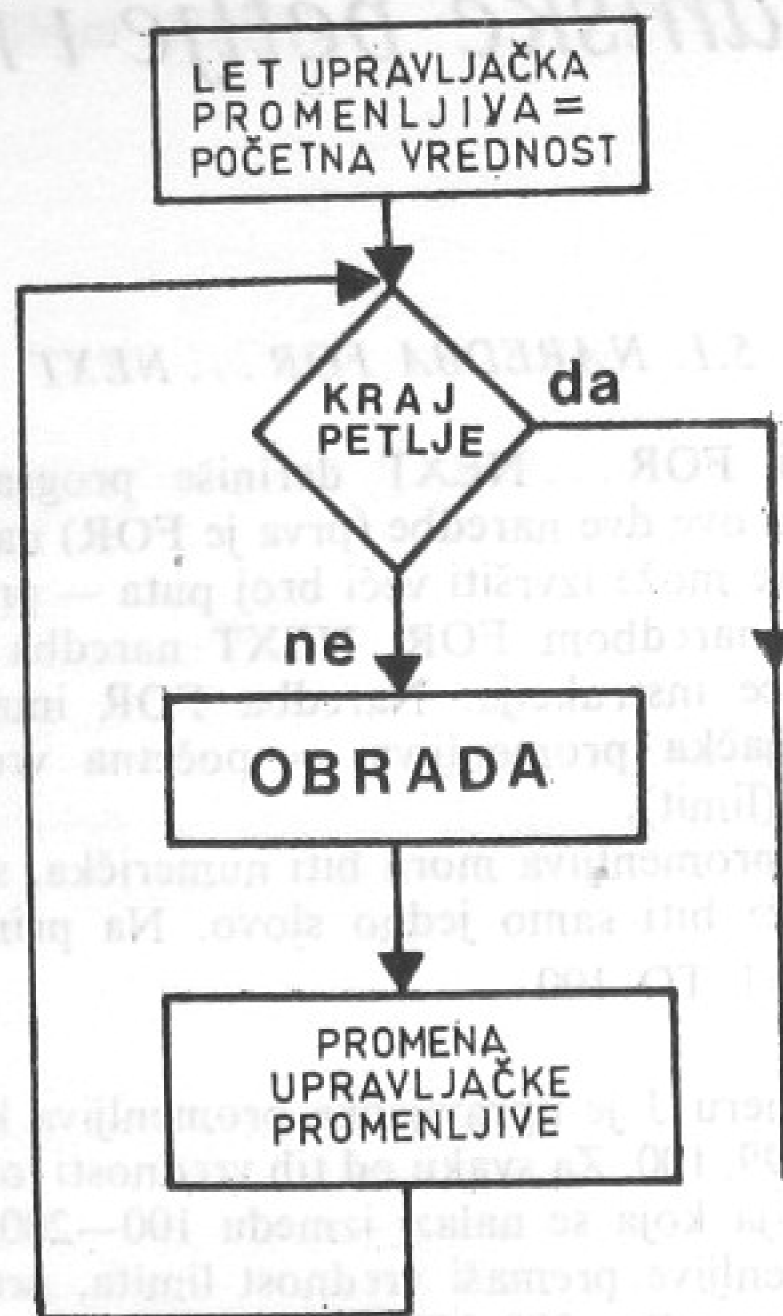
Porast vrednosti upravljačke promenljive J obavlja naredba NEXT. Ovako napisana naredba FOR pretpostavlja da se u svakom sledećem prolazu vrednost upravljačke promenljive povećava za 1. Unesite i izvršite sledeći program:

```
10 FOR A=1 TO 10
20 PRINT TAB 4; A; TAB 8; A*A
30 NEXT A
```

Upravljačka promenljiva ove petlje je promenljiva A, a linija 20 (jedina obuhvaćena petljom) prikazuje vrednosti A i A^2. Limiti indeksa su očigledni: A se menja od 1 do 10.

Pogledajte sledeći primer:

```
10 LET A=8
20 LET B=18
30 FOR C=A TO B
40 PRINT TAB 4; C; TAB 8; C/10 TAB 14; C/A
50 NEXT C
```



Sl. 5.1 — Dijagram odvijanja programske FOR . . . NEXT petlje

Primećujete da su limiti petlje FOR . . . NEXT definisani sadržajima promenljivih A i B. U opštem slučaju limiti petlje mogu biti aritmetički izrazi. Ovu pogodnost često koristimo u raznim slučajevima u praksi. Program koji sledi treba da računa zbir svih prirodnih brojeva počev od nekog prirodnog broja A do nekog B. Vrednosti za A i B se preuzimaju naredbom INPUT.


```
10 REM ZBIR PRIRODNIH BROJEVA U INTERVALU A-B
20 INPUT "UNESITE VREDNOST ZA A (DONJU GRA-
   NICU INTERVALA)"; A$
30 FOR N=1 TO LEN A$
40 IF CODE A$ (N TO N)<48
   OR CODE A$ (N TO N)>57
   THEN BEEP .5,1: GO TO 20
50 NEXT N
60 LET A = VAL A$
70 INPUT "UNESITE VREDNOST ZA B (GORNJU GRA-
   NICU INTERVALA)"; B$
80 FOR N=1 TO LEN B$
90 IF CODE B$ (N TO N)<48
   OR CODE B$ (N TO N)>57
   THEN BEEP .5,1 : GO TO 70
100 NEXT N
110 LET B=VAL B$
120 IF A>B THEN BEEP .5,1 : GO TO 20
130 LET ZBIR = 0
140 FOR N=A TO B
150 LET ZBIR=ZBIR+N
160 NEXT N
170 PRINT "ZBIR PRIRODNIH BROJEVA U INTER-
   VALU"; A; "DO"; B; "JE"; ZBIR
180 STOP
```

Vidimo da su A i B granične vrednosti upravljačke promenljive N, naredba 140, te da ih možemo slobodno birati pri izvršenju programa.

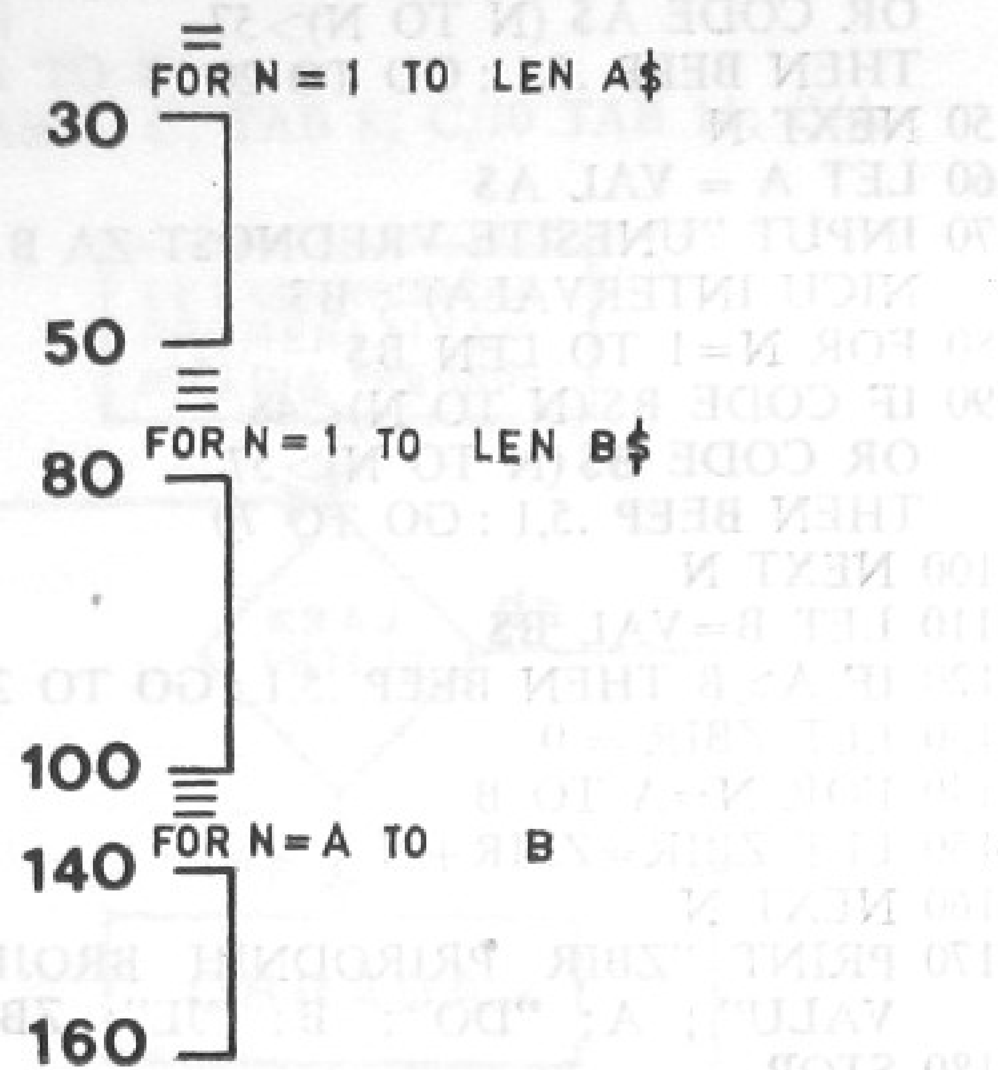
Program se sastoji od tri petlje. Prva petlja služi za kontrolu sadržaja promenljive A. Postupak kontrole je isti kao onaj koji je već objašnjen u prethodnom poglavlju. U slučaju da se unese neki neželjeni znak, oglašava se zvučni signal BEEP, izlazi se iz petlje i zahteva se da se ponovo izvrši operacija ulaza. Promena vrednosti za promenljivu N vrši se "automatski" zato što je ona upravljačka promenljiva. Naredbe FOR . . . NEXT zamenjuju skup od nekoliko naredbi (20, 30, 50 i 60) u programu koji u prethodnom poglavlju vrši kontrolu unosa.

Sledeća petlja vrši istu funkciju kao prethodna s tom razlikom da se odnosi na kontrolu unete vrednosti za promenljivu B.

Treća petlja vrši sabiranje prirodnih brojeva u intervalu od A do B. Pri tome upravljačka promenljiva uzima sledeće vrednosti: A, A+1, A+2, . . . , B.

Nije obavezno da upravljačka promenljiva ima uvek isti naziv u toku programa. Upravljačka promenljiva prve petlje mogla se zvati I, druga J, a treća K. Izmeniti program u tom smislu.

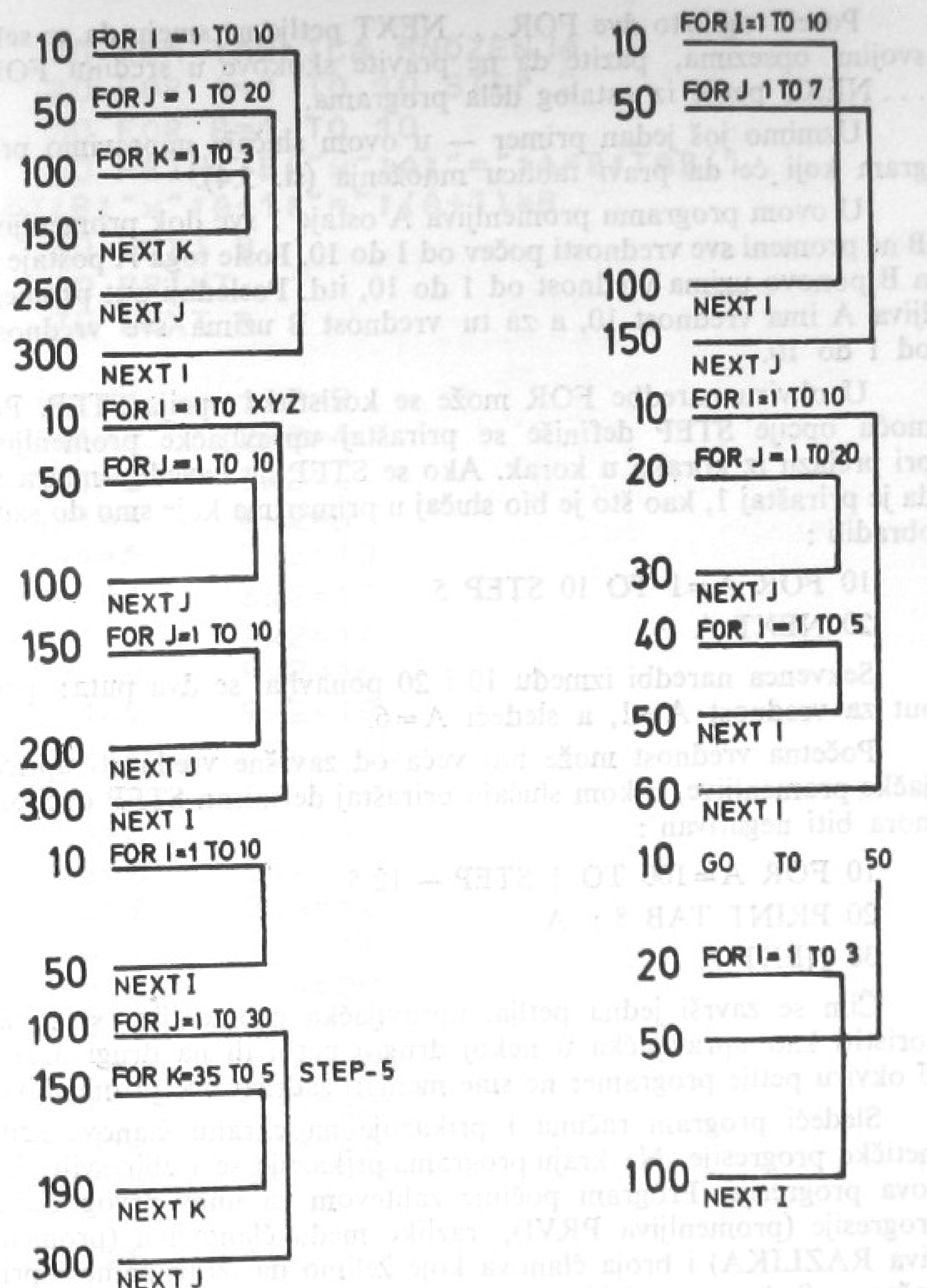
Obratite pažnju da tada u naredbi 40 mora pisati A (I TO I), u naredbi 90 B (J TO J), dok će naredba 150 biti LET ZBIR = ZBIR + K.



Sl. 5.2 — Programske petlje korišćene u programu "Zbir prirodnih brojeva u intervalu A — B"

U okviru sekvence naredbi koje se nalaze u okviru jedne FOR . . . NEXT petlje može se naći i neka druga FOR . . . NEXT petlja. Pri tome moramo biti pažljivi, pa u programu imati petlje koje su potpuno jedna u drugoj ili jedna van druge:

```
5 REM POLOZAJ PETLJI U OVOM PROGRAMU JE
  DOBAR
10 FOR M=0 TO 5
20 FOR N=0 TO M
30 PRINT M;" ";N      } n-petlja   } m-petlja
40 NEXT N
50 PRINT
60 NEXT M
```

Sl. 5.3 — Ispravno i neispravno formirane programske petlje

Pored toga što dve FOR . . . NEXT petlje ne smeju da se seku svojim opsezima, pazite da ne pravite skokove u sredinu FOR- . . . NEXT petlje iz ostalog dêla programa.

Uzmimo još jedan primer — u ovom slučaju napravimo program koji će da pravi tablicu množenja (sl. 5.4).

U ovom programu promenljiva A ostaje 1 sve dok promenljiva B ne promeni sve vrednosti počev od 1 do 10. Posle toga A postaje 2, a B ponovo uzima vrednost od 1 do 10, itd. Poslednji put promenljiva A ima vrednost 10, a za tu vrednost B uzima sve vrednosti od 1 do 10.

U okviru naredbe FOR može se koristiti i opcija STEP. Pomoću opcije STEP definiše se priraštaj upravljačke promenljive pri prelazu iz koraka u korak. Ako se STEP ne navede, smatra se da je priraštaj 1, kao što je bio slučaj u primerima koje smo do sada obradili :

```
10 FOR A=1 TO 10 STEP 5
20 NEXT A
```

Sekvenca naredbi između 10 i 20 ponavlja se dva puta: prvi put za vrednost A=1, a sledeći A=6.

Početna vrednost može biti veća od završne vrednosti upravljačke promenljive, u kom slučaju priraštaj definisan STEP opcijom mora biti negativan :

```
10 FOR A=100 TO 1 STEP — 12.5
20 PRINT TAB 8 ; A
30 NEXT A
```

Čim se završi jedna petlja, upravljačka promenljiva se može koristiti kao upravljačka u nekoj drugoj petlji ili na drugi način. U okviru petlje programer ne sme menjati sadržaj ove promenljive.

Sledeći program računa i prikazuje na ekranu članove aritmetičke progresije. Na kraju programa prikazuje se i zbir svih članova progresije. Program počinje zahtevom za unos prvog člana progresije (promenljiva PRVI), razlike među članovima (promenljiva RAZLIKA) i broja članova koje želimo da izračunamo i prikažemo. Sada nas najviše zanima deo programa koji se realizuje kroz FOR . . . NEXT petlju. U skupu naredbi koje se ponavljaju pod kontrolom FOR . . . NEXT petlje nalazi se i :

```
LET Q=PRVI+L*RAZLIKA
```



```
10 REM TABLICA MNOZENJA
20 FOR A=1 TO 10 STEP 2
30 FOR B=1 TO 10
40 PRINT B;"x";A;"=";A*B;TAB(1
4);B;"x";A+1;"=";(A+1)*B
50 NEXT B
60 PRINT
70 NEXT A
```

```
1x1=1      1x2=2
2x1=2      2x2=4
3x1=3      3x2=6
4x1=4      4x2=8
5x1=5      5x2=10
6x1=6      6x2=12
7x1=7      7x2=14
8x1=8      8x2=16
9x1=9      9x2=18
10x1=10    10x2=20

1x3=3      1x4=4
2x3=6      2x4=8
3x3=9      3x4=12
4x3=12     4x4=16
5x3=15     5x4=20
6x3=18     6x4=24
7x3=21     7x4=28
8x3=24     8x4=32
9x3=27     9x4=36
10x3=30    10x4=40

1x5=5      1x6=6
2x5=10     2x6=12
3x5=15     3x6=18
4x5=20     4x6=24
5x5=25     5x6=30
```

```
6x5=30      6x6=36
7x5=35      7x6=42
8x5=40      8x6=48
9x5=45      9x6=54
10x5=50     10x6=60

1x7=7       1x8=8
2x7=14      2x8=16
3x7=21      3x8=24
4x7=28      4x8=32
5x7=35      5x8=40
6x7=42      6x8=48
7x7=49      7x8=56
8x7=56      8x8=64
9x7=63      9x8=72
10x7=70     10x8=80

1x9=9       1x10=10
2x9=18      2x10=20
3x9=27      3x10=30
4x9=36      4x10=40
5x9=45      5x10=50
6x9=54      6x10=60
7x9=63      7x10=70
8x9=72      8x10=80
9x9=81      9x10=90
10x9=90     10x10=100
```

Sl. 5.4 — Program "Tablica množenja" i rezultati rada programa

gde je L upravljačka promenljive petlje. Pri svakom sledećem izvršavanju navedene naredbe L se povećava za 1, pa ako je PRVI na početku 10, RAZLIKA će tada 10, 12, 14 ... biti prvi članovi progresije (sl. 5.5).

Program računanja brojčane funkcije n! = 1·2·3·4·...·(n—1)·n takođe koristi FOR ... NEXT petlju. Program počinje učitavanjem vrednosti za N koji mora biti prirodan broj. Potom se izračunava i prikazuje izračunata vrednost za NFAK (sl. 5.6).


```
 10 REM ARITMETICKA PROGRESIJA
 20 INPUT "PRVI CLAN PROGRESIJE
JE ";PRVI
 30 INPUT "RAZLIKA JE ";RAZLIKA
 40 INPUT "KOLIKO CLANOVA ZELIT
E ";N
 45 LET ZBIR=PRVI
 50 FOR L=1 TO N-1
 60 LET Q= PRVI+L*RAZLIKA
 70 PRINT L+1;". CLAN JE ";Q
 80 LET ZBIR=ZBIR+Q
 90 NEXT L
100 PRINT "ZBIR CLANOVA 1 DO ";
N;" JE ";ZBIR

PRVI CLAN PROGRESIJE JE 10
RAZLIKA JE 3
KOLIKO CLANOVA ZELITE 4
2. CLAN JE 13
3. CLAN JE 16
4. CLAN JE 19
ZBIR CLANOVA 1 DO  4  JE  58
```

Sl. 5.5 — Program "Aritmetička progresija" i rezultati rada programa

```
 10 LET B$="UNESITE BROJ ZA IZR
ACUNAVANJE FAKTORIJELA "
 20 INPUT (B$);N
 25 IF N<1 THEN STOP
 30 LET NFAK=1
 40 FOR I=1 TO N
 50 LET NFAK=NFAK*I
 60 NEXT I
 70 PRINT N;"! =";NFAK
 80 GO TO 10
```

Sl. 5.6 — Program izračunavanja vrednosti brojčane funkcije n!

Program računa $\frac{n!}{k!\,(n-k)!}$ to na dva načina. U prvoj verziji program se sastoji od tri petlje, od kojih prva računa n!, druga k!, a treća (n—k)!. Izračunavanje faktorijela vrši se na isti način kao u prethodnom programu. U drugoj verziji koristi se samo jedna petlja, u kojoj se računa vrednost sve tri faktorijelne funkcije (sl. 5.7, 5.8 i 5.9).

```
 1 REM IZRACUNAVANJE N NAD K
 10 REM N!/(K!*(N-K)!)
 15 INPUT "N=";N
 20 INPUT "K=";K
 25 IF K>N THEN STOP
 30 LET NFAK=1
 35 REM PROGRAMSKA PETLJA ZA IZ
RACUNAVANJE  N!
 40 FOR X=1 TO N
 50 LET NFAK=NFAK*X
 60 NEXT X
 100 LET KFAK=1
 105 REM PROGRAMSKA PETLJA ZA IZ
RACUNAVANJE  K!
 110 FOR X=1 TO K
 120 LET KFAK=KFAK*X
 130 NEXT X
 200 LET NMANJEKFAK=1
 205 REM PROGRAMSKA PETLJA ZA IZ
RACUNAVANJE  (N-K)!
 210 FOR X=1 TO N-K
 220 LET NMANJEKFAK=NMANJEKFAK*X
 230 NEXT X
 300 PRINT "REZULTAT JE ";NFAK/(
KFAK*NMANJEKFAK)
```

Sl. 5.7 — Program "Izračunavanje N nad K" (prva verzija)

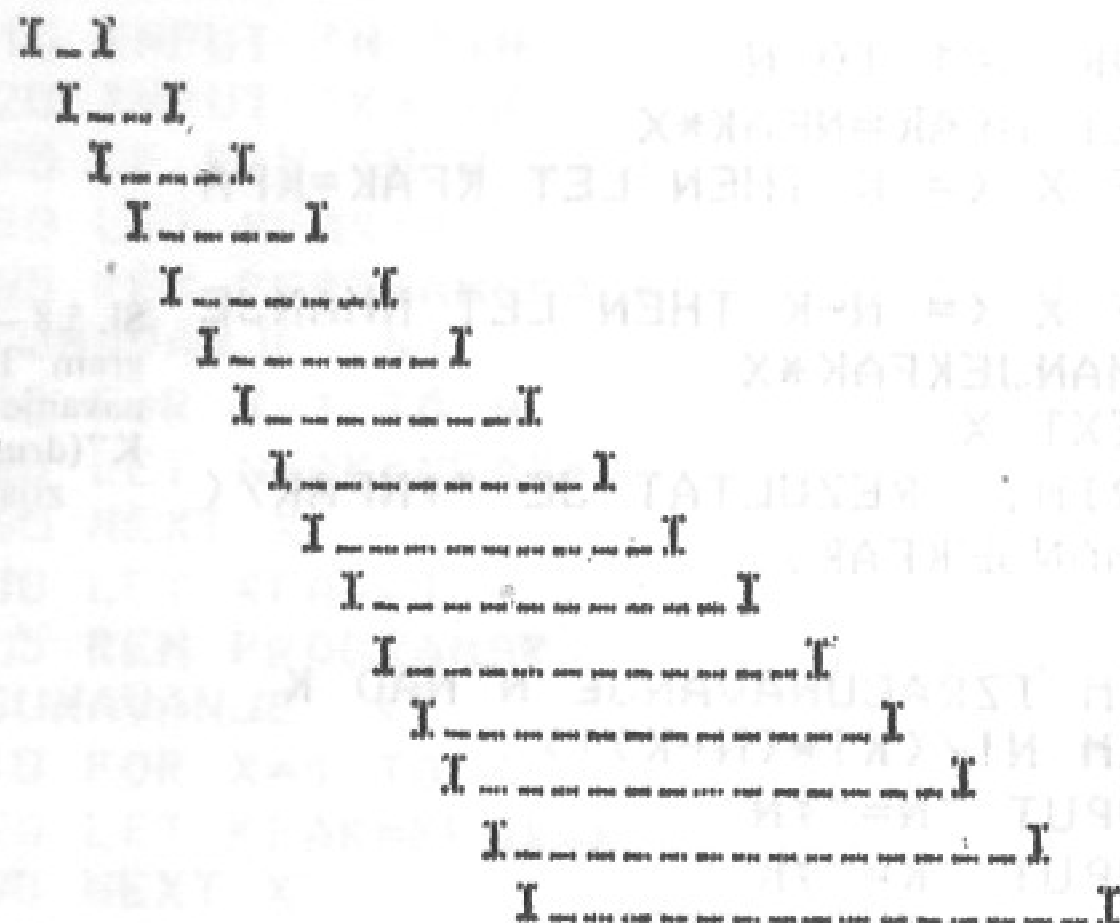
Sledeći program se sastoji od jedne dvostruke FOR . . NEXT petlje. Upravljačka promenljiva u tzv. spoljnoj petlji je I, a u unutrašnjoj J. Primetite da je limit promenljive J u funkciji od I (sl. 5.10).


```
 1 REM IZRACUNAVANJE N NAD K
 10 REM N!/(K!*(N-K)!)
 15 INPUT "N=";N
 20 INPUT "K=";K
 25 IF K>N THEN STOP
 30 LET NFAK=1
 40 LET KFAK=1
 50 LET NMANJEKFAK=1
 55 REM ZRACUNAVANJE N! K! I (N
-K)!
 60 FOR X=1 TO N
 70 LET NFAK=NFAK*X
 80 IF X <= K THEN LET KFAK=KFA
K*X
 90 IF X <= N-K THEN LET NMANJE
KFAK=NMANJEKFAK*X
 100 NEXT X
 300 PRINT "REZULTAT JE ";NFAK/(
KFAK*NMANJEKFAK)
```

Sl. 5.8 — Program "Izračunavanje N nad K" (druga verzija)

```
 1 REM IZRACUNAVANJE N NAD K
 10 REM N!/(K!*(N-K)!)
 15 INPUT "N=";N
 20 INPUT "K=";K
 25 IF K>N THEN STOP
 30 LET REZ=1
 40 LET Z=N-K
 50 IF K > N-K THEN LET Z=K
 51 LET Z=Z+1
 52 REM KORISTIMO SKRACENI POST
UPAK ZA IZRACUNAVANJE
 56 REM N*(N-1)*.....*(N-K+1)/(
K!) ILI
 57 REM N*(N-1)*.....*(K+1)/(N-
K)!
 60 FOR X=N TO Z STEP -1
 70 LET REZ=REZ*X/(N-X+1)
 100 NEXT X
 300 PRINT "REZULTAT JE ";REZ
```

Sl. 5.9 — Program "Izračunavanje N nad K" (druga verzija — skraćeni postupak)



```
10 REM PROGRAMSKA PETLJA
20 FOR I=1 TO 15
30 PRINT TAB (I);"I";
40 FOR J=1 TO I
50 PRINT "-";
60 NEXT J
70 PRINT "I"
80 NEXT I
```

Sl. 5.10 — Program "Programska petlja" i rezultati rada programa

5.2. PROGRAMSKE RUTINE (NAREDBA GO SUB . . . RETURN)

Naredba GO SUB iza koje sledi broj naredbe često se koristi kod programiranja složenijih programa u BASIC-u. Kada se pri izvršenju programa naiđe na naredbu GO SUB, vrši se automatski prelaz na naredbu čiji broj naredbe je uz GO SUB. Tako naredba:

```
20 GO SUB 800
```

obezbeđuje prelaz na naredbu 800. Posle ove operacije prelaza izvršava se naredba 800, i naredbi koje je slede sve dok se ne dođe

do naredbe RETURN. Kada se naiđe na naredbu RETURN, izvršiće se ponovo automatski prelaz, ovoga puta na naredbu koja sledi naredbu GO SUB :

```
10 REM GO SUB
20 LET A=1
30 LET B=4
40 GO SUB 80
50 PRINT "C= ";C;"D= ";D
70 STOP
80 LET C=A+B
90 LET D=C+12
100 RETURN
```

Pre nego što se dođe do naredbe 40, promenljive A i B imaju definisane sadržaje 1 i 4. Naredbom GO SUB prvo se prelazi na naredbu 80. (Ovaj je prelaz isti kao prelaz koji se realizuje naredbom GO TO.) Potom se urade naredbe 80 i 90 kojima se izračunavaju vrednosti za sadržaje promenljivih C i D. Kada se dođe do naredbe 100 RETURN, vrši se prelaz unazad na naredbu koja sledi naredbu GO SUB. To je naredba 50 kojom se vrši prikaz izračunatih vrednosti za C i D. Da smo prelaz na naredbu pod brojem 80 vršili naredbom GO TO 80, umesto naredbe 100 RETURN morala bi postojati naredba 100 GO TO 50, što u ovome primeru ne bi predstavljalo problem.

Skup naredbi koje se pozivaju na izvršenje naredbom GO SUB često se naziva programskom rutinom.

Jedna programska rutina se može pozivati sa više mesta u programu. Ako ne bi postojao mehanizam automatskog povratka koji se obezbeđuje pri izvršenju naredbe RETURN, morali bismo pri izvršenju programske rutine sami da obezbedimo mehanizam povratka. To se čini naredbom GO TO, ali pazite! Potrebno je voditi evidenciju sa koga mesta se u programu rutina pozvala:

```
10 REM GO SUB          10 REM GO TO
20 . . .               20. . .
30 GO SUB 100          29 LET POVRATAK = 40
40 . . .               30 GO TO 100
50 GO SUB 100          40 . . .
60 . . .               49 LET POVRATAK = 60
70 STOP                50 GO TO 100
100 . . .              60 . . .
```

```
110 . . .
120 RETURN
```

```
70 STOP
100 . . .
110 . . .
120 IF POVRATAK = 40
    THEN GO TO 40
130 IF POVRATAK = 60
    THEN GO TO 60
```

Vođenje evidencije o mestu u programu sa kojega smo pozivali programsku rutinu čini rutinu potpuno zavisnom od programa. Sa druge strane, programske rutine koje se formiraju sa GO SUB . . . RETURN gotovo su nezavisne od programa, te se mogu prebacivati iz jednog programa u drugi. Jedino treba voditi računa o brojevima naredbi.

Sledeći program učitava seriju brojeva $X_1, X_2, \ldots X_n$, i prema izboru računa:

1. aritmetičku sredinu,
2. geometrijsku sredinu,
3. harmonijsku sredinu,
4. standardnu devijaciju,

a potom prikazuje na ekranu izračunatu vrednost. Vrednosti za X_i ima neodređen broj, a pošto se unese poslednja vrednost za X_i, na zahtev za ponovni unos tipkajte KRAJ:

$$\text{Aritmetička sredina} = \frac{\sum_{i=1}^{N} X_i}{N}$$

$$\text{Geometrijska sredina} = \sqrt[N]{X_1 \cdot X_2 \cdot X_3 \cdots X_n}$$

$$\text{Harmonijska sredina} = \frac{N}{\sum_{i=1}^{N} \frac{1}{X_i}}$$

$$\text{Standardna devijacija} = \sqrt[2]{\frac{\sum_{i=1}^{N} X_i^2 - \frac{\left(\sum_{i=1}^{N} X_i\right)^2}{N}}{N-1}}$$

Program ćemo raditi deo po deo. Prvo ćemo napraviti deo programa koji računa aritmetičku sredinu.

Da bi se došlo do aritmetičke sredine, potrebno je sabrati sve X_i i prebrojati ih. Za te operacije koristićemo dve promenljive SUMAX i N. Obe promenljive menjaju svoj sadržaj sabiranjem, pa će im početne vrednosti biti jednake nuli:

```
900 REM ARITMETICKA SREDINA
1000 PRINT "UNESITE BROJEVE ZA KOJE"
1010 PRINT "SE RACUNA ARITMETICKA SREDINA"
1020 PRINT "U SLUCAJU KRAJA UNESITE KRAJ"
1030 LET N=0 : LET SUMAX = 0
1040 INPUT "BROJ JE : " ; X$
1050 IF X$ = "KRAJ" THEN GO TO 1090
1060 LET N = N + 1
1070 LET SUMAX = SUMAX + VAL X$
1080 GO TO 1040
1090 PRINT "ARITMETICKA SREDINA JE"; SUMAX/N
```

Program možete da unesete i odmah da ispitate da li dobro radi.

Sledeći korak je izračunavanje geometrijske sredine. Da bi se izračunala geometrijska sredina, potrebno je pomnožiti sve X_i i prebrojati ih. Za te operacije koristićemo promenljive PROX i N. Početna vrednost promenljive PROX je 1, a promenljive N je nula:

```
1900 REM GEOMETRIJSKA SREDINA
2000 PRINT "UNESITE BROJEVE ZA KOJE"
2010 PRINT "SE RACUNA GEOMETRIJSKA SREDINA"
2020 PRINT "U SLUCAJU KRAJA UNESITE KRAJ"
2030 LET PROX = 1 : LET N=0
2040 INPUT "BROJ JE" ; X$
2050 IF X$ = "KRAJ" THEN GO TO 2090
2060 LET N = N + 1
2070 LET PROX = PROX * VAL X$
2080 GO TO 2040
2090 PRINT "GEOMETRIJSKA SREDINA JE:";
     PROX ↑ (1/N)
```

Treći deo programa izračunava harmonijsku sredinu. Da bismo je izračunali, potrebno je da saberemo recipročne vrednosti svih X_i i da ih prebrojimo. Za te operacije koristimo promenljive RECX i N. Obe su na početku programa jednake nuli:

```
2900 REM HARMONIJSKA SREDINA
3000 PRINT "UNESITE BROJEVE ZA KOJE"
3010 PRINT "SE RACUNA HARMONIJSKA SREDINA"
3020 PRINT "U SLUCAJU KRAJA UNESITE KRAJ"
3030 LET RECX = 0 : LET N=0
3040 INPUT "BROJ JE"; X$
3050 IF X$="KRAJ" THEN GO TO 3090
3060 LET N=N+1
3070 LET RECX = RECX + (1/VAL X$)
3080 GO TO 3040
3090 PRINT "HARMONIJSKA SREDINA JE" ;
     N/RECX
```

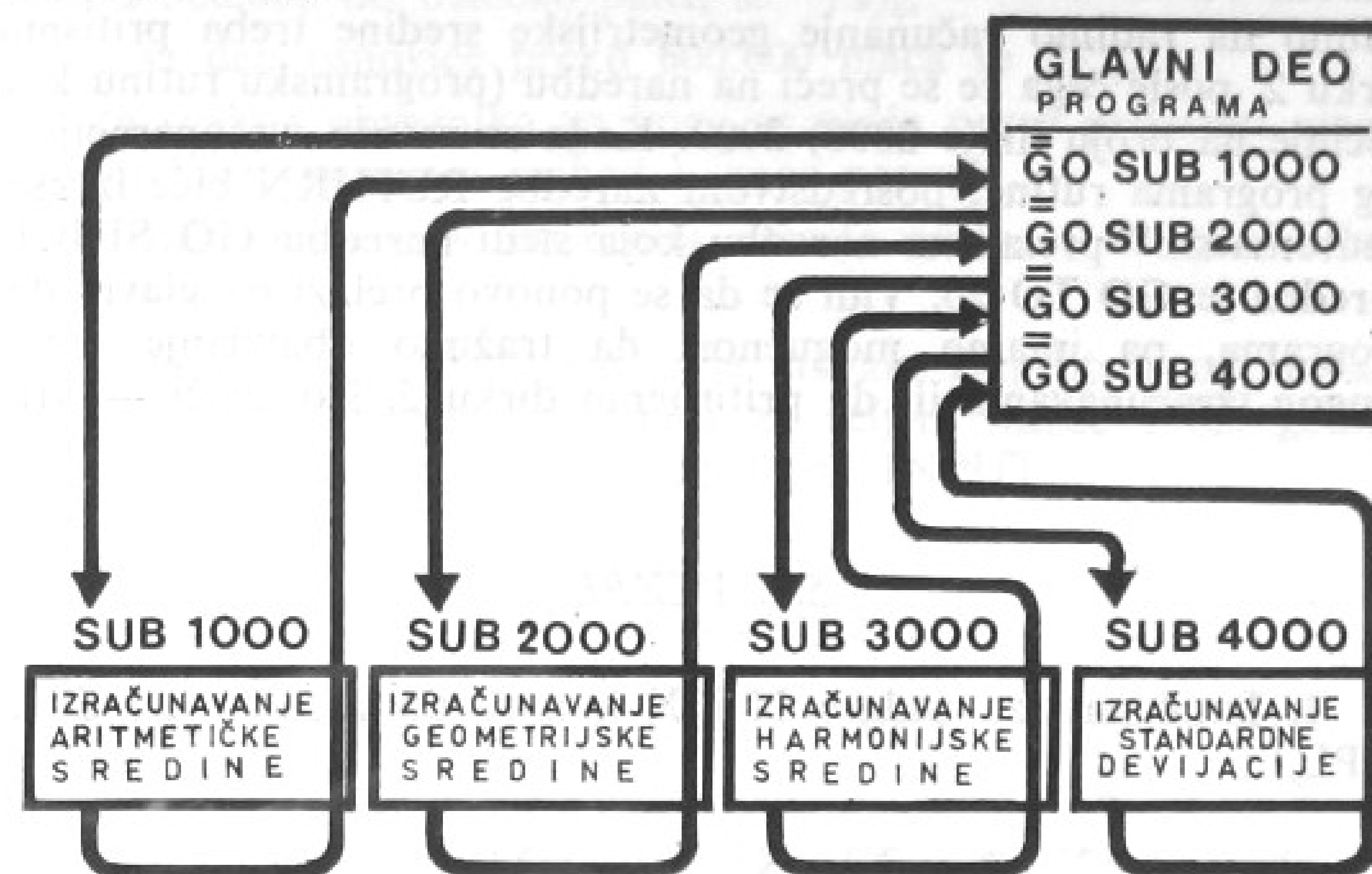
Poslednje izračunavanje koje ćemo izvršiti jeste računanje standardne devijacije. Za izračunavanje ove veličine potrebno je izračunati zbir svih X_i, zbir kvadrata svih X_i, kao i njihov broj. Prvu promenljivu zvaćemo SUMAX, sledeću SUKVX, a treću N. Sve tri promenljive su na početku programa jednake nuli:

```
3900 REM STANDARDNA DEVIJACIJA
4000 PRINT "UNESITE BROJEVE ZA KOJE"
4010 PRINT "SE RACUNA STANDARDNA DEVIJACIJA"
4020 PRINT "U SLUCAJU KRAJA UNESITE KRAJ"
4030 LET SUMAX=0:LET SUKVX=0:LET N=0
4040 INPUT "BROJ JE" ; X$
4050 IF X$="KRAJ" THEN GO TO 4100
4060 LET N = N + 1
4070 LET SUMAX = SUMAX + VAL X$
4080 LET SUKVX = SUKVX + VAL X$ * VAL X$
4090 GO TO 4040
4100 PRINT "STANDARDNA DEVIJACIJA JE";
SQR ((SUKVX – SUMAX * SUMAX/N)/(N – 1))
```

Svaki od navedena četiri programa predstavlja u ovom slučaju celinu potpuno nezavisnu od ostalih. Formirajmo od njih programske rutine. Sve što je potrebno da uradimo, to je da dodamo na kraj svake rutine naredbu RETURN. Tako ćemo u program dodati:

```
1120 RETURN
2120 RETURN
3120 RETURN
4120 RETURN
```


Na taj način su delovi programa i formalno postali programske rutine (sl. 5.11).



Sl. 5.11 — Programske rutine

Ostalo je još da završimo deo programa koji vrši izbor među otpisanim rutinama. Tako koncipiran deo programa često se zove meni :

```
 10 REM GLAVNI DEO PROGRAMA MENI
 20 PRINT "IZABERITE STA ZELITE DA RACUNATE"
 30 PRINT "1 – ARITMETICKA SREDINA"
 40 PRINT "2 – GEOMETRIJSKA SREDINA"
 50 PRINT "3 – HARMONIJSKA SREDINA"
 60 PRINT "4 – STANDARDNA DEVIJACIJA"
 70 PRINT "5 – KRAJ"
 80 LET A$ = INKEY$
 90 IF A$<"1" OR A$>"5"
    THEN GO TO 80
100 LET A = VAL A$
110 IF A = 5 THEN STOP
120 GO SUB A * 1000
130 GO TO 20
```

Naredbom 120 izabira se jedna od četiri programske rutine. Adresa rutine se određuje aritmetičkim izrazom A * 1000. Ako želimo da radimo računanje geometrijske sredine treba pritisnuti dirku 2, posle čega će se preći na naredbu (programsku rutinu koja počinje na broju linije 2000) 2000. Kada se završe sve operacije iz tog programa rutine, posredstvom naredbe RETURN biće izvršen „automatski" prelaz na naredbu koja sledi naredbu GO SUB. Ta naredba je GO TO 20. Vidi se da se ponovo prelazi na glavni deo programa, pa imamo mogućnost da tražimo obavljanje nekog programa, pa imamo mogućnost da tražimo obavljanje nekog drugog izračunavanja ili da pritisnemo dirku 5, što znači — kraj.

5.3. VEŽBE

1. Izračunati za zadato N i X koji se preuzimaju naredbom INPUT:

a) $1+X-X^2+X^3-X^4+\cdots\pm X^n$

b) $1+2X+3X^2+4X^3+\cdots+nX^{n-1}$

c) $X-\frac{1}{3}X^3+\frac{1}{5}X^5-\frac{1}{7}X^7+\cdots$

d) $X+\frac{X^2}{2!}+\frac{X^3}{3!}+\frac{X^4}{4!}+\cdots+\frac{X^n}{n!}$

2. U poreskom uredu vodi se evidencija o obveznicima koji treba da plate porez. Za svakog obveznika postoje tri podatka:

— matični broj,
— ostvaren prihod,
— broj izdržavanih članova porodice.

Iznos na koji se ne plaća porez je 400.000 uvećan za po 50.000 za svakog izdržavanog člana porodice.

Razlika ostvarenog prihoda i iznosa na koji se ne plaća porez je poreska osnovica.

Na deo osnovice do 300.000 plaća se 5%,

od 300.000 do 600.000 plaća se 15%,

a za deo osnovice preko 600.000 plaća se 35%.

Za svakog obveznika za koga se plaća porez prikazati njegov matični broj, ostvareni prihod i iznos poreza.

Kada se obrade svi obveznici, umesto vrednosti za matični broj unosi se nula.

3. Prvi januar 1985. godine je utorak. Napišite program koji na ekranu vrši prikaz kalendara za željeni mesec 1985. godine. Zahtev za mesec prihvatajte naredbom INPUT.

JANUAR 1985.

NED	PON	UTO	SRE	CET	PET	SUB
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15				

Proširite program tako da radi za bilo koju godinu.

4. Napravite program koji prikazuje sve proste brojeve počev od 1 do zadatog N (koje se prihvata naredbom INPUT).

5. Poruka ispisana Morzeovim znacima biće ulaz u vaš program. Učitavaće se uvek po jedan Morzeov znak (jednom naredbom INPUT). Kada se unese cela reč, uneće se jedan blanko znak. Program treba da čita poruku, dekodira i prikaže je na ekranu:

A	· —	J	· — — —	S	· · ·
B	— · · ·	K	— · —	T	—
C	— · — ·	L	· — · ·	U	· · —
D	— · ·	M	— —	V	· · · —
E	·	N	— ·	W	· — —
F	· · — ·	O	— — —	X	— · · —
G	— — ·	P	· — — ·	Y	— · — —
H	· · · ·	Q	— — · —	Z	— — · ·
I	· ·	R	· — ·		

6. Napišite program koji na ekranu prikazuje Paskalov trougao. Članovi trougla su funkcije $\binom{n}{k}$, kao što je prikazano:

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!}$$

$$\binom{0}{0}$$

$$\binom{1}{0} \quad \binom{1}{1}$$

$$\binom{2}{0} \quad \binom{2}{1} \quad \binom{2}{2}$$

$$\binom{3}{0} \quad \binom{3}{1} \quad \binom{3}{2} \quad \binom{3}{3}$$

$$\binom{4}{0} \quad \binom{4}{1} \quad \binom{4}{2} \quad \binom{4}{3} \quad \binom{4}{4}$$

:

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

:

7. Napišite program koji čita arapske, a prikazuje odgovarajuće rimske brojeve. Na primer:

Ulaz	*Prikaz*
1	I
21	XXI
1984	MCMLXXXIV

Rimski brojevi imaju sledeće vrednosti:

M	1000	X	10
D	500	V	5
C	100	I	1
L	50		



Promenljive sa strukturom niza

6.1. NAREDBE READ, DATA, RESTORE

Videli smo da je jedan od mogućih načina da se dodeli sadržaj nekoj promenljivoj — naredba INPUT. Ova naredba zahteva komunikaciju sa korisnikom za vreme izvršenja programa. To može imati i svoje loše strane, kao što je: stalno unošenje većeg broja podataka pri izvršenju, te mogućnosti da se pri unosu vrše greške koje se mogu otkriti tek pri izvršenju programa. Zato je korisno da svi podaci koji su konstantni za program budu u njega ugrađeni. To se postiže parom naredbi READ i DATA.

Naredbu READ prati lista naziva promenljivih međusobno odvojenih zarezima, na primer:

READ A, B, C, D

Ona radi slično naredbi INPUT, s tom razlikom što se posebne vrednosti za promenljive ne unose preko tastature, već se preuzimaju iz samog programa gde su te vrednosti definisane u okviru DATA naredbe.

Naredbu DATA prati lista konstanti odvojenih zarezima, kao:

DATA 1, 5, 7, 9

DATA "JANUAR", "FEBRUAR"

Naredba DATA se može naći u bilo kom delu programa. To je stoga što naredba DATA ne predstavlja naredbu izvršnog tipa, već samo definiciju uređenog skupa konstanti koje postaju aktuelne tek kad se aktivira neka od naredbi READ. Najčešće se sve naredbe

DATA stave na kraj programa. Veoma je bitan redosled konstanti u okviru svih DATA naredbi. Prva READ naredba dodeljuje vrednosti promenljivim (koje stoje u listi koja sledi naredbu READ) u redosledu kojim su one navedene u prvoj DATA naredbi:

```
20 READ A, B
60 DATA 10, 11, 17, 27, 36, 7
```

Kada se prvi put izvrši naredba 20, sadržaj promenljivih će biti A=10, B=11; posle drugog izvršenja naredbe READ biće A=17, B=27, a treći put A=36, B=7. Potpuno ista situacija biće u slučaju da smo umesto naredbe:

```
60 DATA 10, 11, 17, 27, 36, 7
```

napisali više DATA naredbi:

```
60 DATA 10, 11, 17
70 DATA 27, 36, 7
```

ili

```
10 DATA 10
20 DATA 11
30 DATA 17
120 DATA 26
130 DATA 36
140 DATA 7
```

Sve DATA koje se nađu u programu možemo smatrati za jednu u čijoj listi konstanti se nalaze sve konstante iz svih DATA naredbi u programu, i to u redosledu kojim su DATA naredbe napisane u programu i u redosledu posebnih listi:

```
10 DATA 10
20 READ A, B, C
30 DATA 12, 14
40 PRINT A, B, C
```

Treba paziti da se pri korišćenju READ naredbe ne prekorači DATA lista, jer to prouzrokuje grešku. Promenimo naredbu 20 u prethodnom programu u:

```
20 READ A, B, C, D
```

i izvršimo program. Vidimo da će računar javiti poruku o grešci:

E OUT OF DATA

Korišćenje informacije u DATA listi često je deo neke FOR . . . NEXT petlje:


```
10 FOR A=1TO 10
20 READ B
30 DATA 2, 4, 6, 18
40 DATA 7, 4, 12, 15, 17, 19, 21, 13
50 PRINT A; " ";B
60 NEXT A
```

Ovaj program koristi FOR . . . NEXT petlju kojom se komanduje izvršenje naredbi 20—50, 10 puta. Naredba 50 prikazuje sadržaj promenljive A koja upravlja petljom, i odgovarajuće konstante iz liste DATA (sadržaj promenljive B). Vidimo da u posebnom slučaju sve konstante iz liste ne moraju biti upotrebljene (21, 13).

Naredba DATA može sadržati i nenumeričke konstante:

```
10 READ A$
20 PRINT A$
30 READ B$
40 PRINT B$
50 READ C$
60 PRINT C$
70 DATA "SRECAN", "TI", "RODJENDAN"
```

Pogledajte ovaj program. Da li su nam ovde neophodne tri promenljive A, B, C? Prepravite program tako da koristite FOR . . . NEXT petlju!

Napravimo sada dva programa koji kombinuju korišćenje naredbi INPUT i READ/DATA. Prvi program traži pri izvođenju ime vašeg prijatelja i praznik koji mu čestitate — kao ulaz (INPUT), a potom prikazuje čestitku na ekranu:

```
10 REM CESTITKA
20 INPUT "IME PRIJATELJA" ; I$
30 INPUT "CESTITKA" ; C$
40 CLS
50 READ A$
60 IF A$ = "1" THEN LET A$ = "DRAGI " + I$
70 IF A$ = "2" THEN LET A$ = C$
80 PRINT A$
90 IF A$ = "PERA" THEN STOP
100 GO TO 50
110 DATA "1", "2", "ŽELI TI", "PERA"
```

Šta ćemo dobiti na ekranu?
Ime prijatelja? SLOBODAN
Čestitka? SRECAN RODJENDAN

Potom se obriše ekran, što je posledica naredbe CLS, pa na ekranu izlazi sledeće:

DRAGI SLOBODAN
SRECAN RODJENDAN
ŽELI TI
PERA

Naravno, vi ćete, kada budete unosili program, umesto PERA staviti vaše ime, a vaše prijatelje ćete zamoliti da se ne ljute, jer kompjuter još uvek nije dobro naučio padeže. Drugi program traži pri izvođenju da unesete neki datum u obliku DD, MM, GG, kao ulaz (INPUT), a potom prikazuje na ekranu isti datum, ali sa nazivom meseca ispisanim slovima:

```
10 REM MESECI
20 INPUT "UNESITE DAN U OBLIKU DD"; DD
30 INPUT "UNESITE MESEC U OBLIKU MM"; MM
35 LET MM = INT MM
40 IF MM<1OR MM>12 THEN GO TO 30
50 INPUT "UNESITE GODINU U OBLIKU GG"; GG
60 FOR I=1 TO 12
70 READ A$
80 IF I=MM THEN LET B$=A$
90 NEXT I
100 PRINT "DATUM JE";DD;".";B$;".19";GG
110 DATA "JANUAR", "FEBRUAR", "MART"
120 DATA "APRIL", "MAJ", "JUNI"
130 DATA "JULI", "AVGUST", "SEPTEMBAR"
140 DATA "OKTOBAR", "NOVEMBAR", "DECEMBAR"
```

Postoji mogućnost da se vrše skokovi na određene DATA naredbe, i to korišćenjem naredbe RESTORE. Iza naredbe RESTORE piše se broj naredbe (linije). Kada se izvrši naredba RESTORE, sledeća naredba READ koja se izvršava preuzeće vrednosti iz one naredbe DATA koja se nalazi na, ili je prva ispod navedenog broja naredbe uz RESTORE:

```
10 READ A, B
20 PRINT A, B
30 RESTORE 10
40 READ C, D
50 PRINT C, D
60 DATA 25, 26
```


BROJ NAREDBE	DATA	READ A,B,C,D,E,F,G	RESTORE 100	READ A,B,C,D,E,F,G	RESTORE 80	READ A
90	1	A = 1			POKAZIVAČ ZA SLEDEĆI READ	A = 1
	2	B = 2				POKAZIVAČ ZA SLEDEĆI READ
	3	C = 3				
100	4	D = 4	POKAZIVAČ ZA SLEDEĆI READ	A = 4		
	5	E = 5		B = 5		
	6	F = 6		C = 6		
110	7	G = 7		D = 7		
	8	POKAZIVAČ ZA SLEDEĆI READ		E = 8		
	9			F = 9		
	10			G =10		
				POKAZIVAČ ZA SLEDEĆI READ		

Sl. 6.1 — Ilustracija rada naredbe RESTORE

Prva READ naredba preuzima obe vrednosti iz DATA liste, te je A=25, B=26, i time je DATA lista iscrpena. Naredba RESTORE čini podatke iz DATA naredbe ponovo upotrebljivim, te sledeći READ sada može ponovo da preuzme vrednost iz DATA liste, te PRINT pokazuje C=25, D=26. Primetite da je uz RESTORE mogao da stoji i neki drugi broj naredbe (linije)⩽60. Obrišite liniju 30 i ponovo pustite program da radi. Sada će se javiti već pomenuti komentar o greški koja kaže da je DATA lista već iskorišćena.

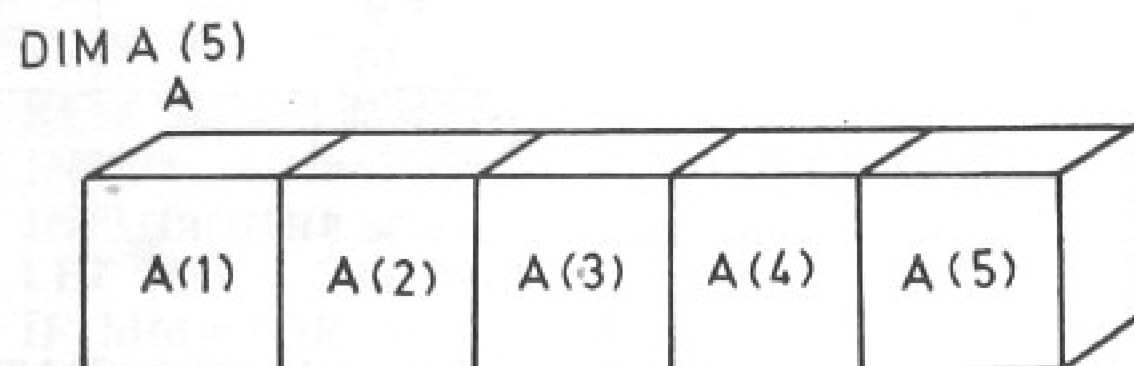
Sledeći program takođe demonstrira upotrebu naredbe RESTORE:

```
 10 READ A, B, C, D, E, F, G
 20 PRINT A;B;C;D;E;F;G
 30 RESTORE 100
 40 READ A,B,C,D,E,F,G
 50 PRINT A,C,D,E,F,G
 60 RESTORE 80
 70 READ A
 80 PRINT A,B,C
 90 DATA 1,2,3
100 DATA 4,5,6
110 DATA 7,8,9,10
```

6.2. PROMENLJIVE SA STRUKTUROM NIZA

BASIC Spectruma dozvoljava i rad sa promenljivim koje imaju strukturu niza. Pod promenljivim koje imaju strukturu niza podrazumevamo skup promenljivih koje (sve) imaju iste nazive, a među sobom se razlikuju po vrednosti indeksa. Sve promenljive moraju biti istog tipa.

Veoma često se skupovi ovih promenljivih zovu tabele ili matrice, pri čemu pojam matrice ne treba mešati sa odgovarajućim pojmom iz matematike.



Sl. 6.2 — Jednodimenzioni niz A

Prvi element niza A je A(1), drugi A(2), . . . itd.

Da bi se moglo raditi sa nizovima, u programu je neophodno dati informaciju o tome koliko jedan niz ima elemenata kako bi se za elemente niza mogao rezervisati odgovarajući prostor. To se čini naredbom DIM iza koje se pored naziva promenljive u zagradi piše najveći broj elemenata koje niz ima.

DIM A(5) definiše niz koji se zove A i koji se sastoji od 5 elemenata A(1), . . . , A(5). Indeksne promenljive, slično kao i upravljačka promenljiva kod FOR . . . NEXT petlje, mogu imati naziv koji se sastoji samo iz jednog slova. Naredba DIM A takođe briše svaki niz koji se zove A (ako je postojao), a svim elementima niza dodeljuje sadržaj 0.

Sa promenljivima koje imaju strukturu niza mogu se raditi sve operacije kao i sa promenljivim koje smo do sada koristili, a koje se nazivaju skalarnim :

```
PRINT A(3)
LET A(2)=A(1)↑2+A(5)
INPUT A(5)
itd.
```


Indeks uz promenljivu tipa niza može, osim konstante, da bude i promenljiva, odnosno neki aritmetički izraz, koji se svodi na celobrojnu vrednost. Ako napišemo, na primer:

```
LET A(2) = A(I)+1
```

to znači da drugom elementu niza A dodeljujemo novu vrednost koja je zbir jedinice i sadržaja I-tog elementa niza A. O kom je elementu reč zavisi isključivo od sadržaja promenljive I u momentu izvršenja navedene naredbe:

25 LET I=2	25 LET I=4
26 LET A(2) = A(I)+1	26 LET A(2) = A(I)+1
A(I) je drugi element niza	A(I) je četvrti element niza

Pošto se za struktuiranje skupa promenljivih u niz najčešće odlučujemo zbog mogućnosti da ih obrađujemo na unificiran način, neretko se uz niz pojavljuju naredbe FOR . . . NEXT:

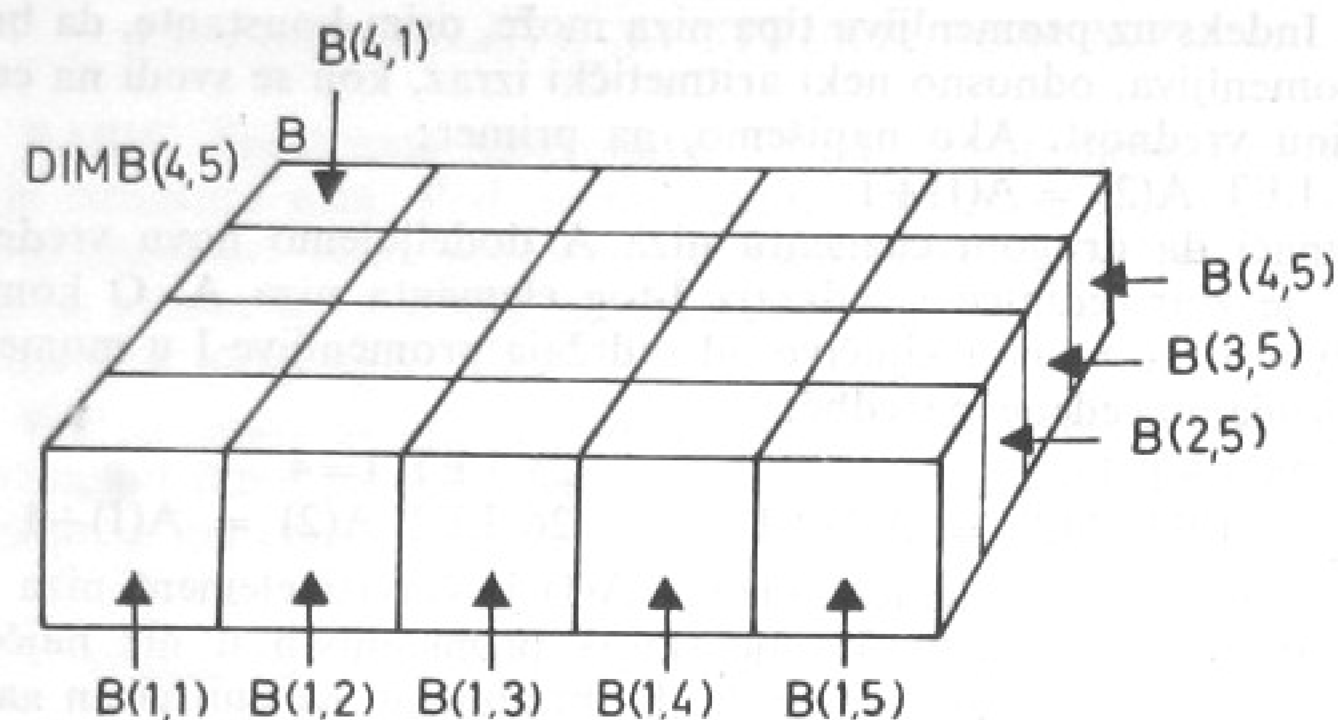
```
10 DIM A(5)
20 FOR I=1 TO 5
30 READ A(I)
40 NEXT I
50 DATA 1, 2, 10, 12, 17
60 PRINT A(1);A(2);A(3);A(4);A(5);A(1)+A(3)
```

U ovom zadatku koristili smo I upravljačku promenljivu FOR . . . NEXT petlje kao indeks niza A. U okviru petlje elementima niza dodeljuju se vrednosti definisane u naredbi DATA. U programu elemente niza koristimo kao: indeksirane promenljivom, indeksirane konstantom ili u okviru aritmetičkog izraza.

Niz A koji smo do sada koristili je jednodimenzioni niz. Kod dvodimenzionog niza potrebne su nam vrednosti dva indeksa da bi se odredila promenljiva unutar niza. Takođe, u okviru DIM se daje maksimalna vrednost po svakom indeksu. DIM B(4,5) znači definisanje dvodimenzionog niza koji ima četiri vrste i pet kolona i koji se sastoji od ukupno 4*5=20 elemenata (promenljivih), indeksiranih na sledeći način:

B(1,1), B(1,2), . . . , B(1,5)
B(2,1), B(2,2), . . . , B(2,5)
.
.
.
B(4,1), B(4,2), . . . , B(4,5)

Slično je i za trodimenzioni niz. Uz njega ide sledeća definicija dimenzije DIM C(2,3,6). Niz se sastoji od 2*3*6=36 elemenata tipa C(p,q,r).



Sl. 6.3 — Dvodimenzioni niz B

Sledi nekoliko programa kod kojih je zajedničko formiranje i prikaz sadržaja nizova različite dimenzije (sl. 6.4, 6.5 i 6.6).

Napomenimo da kod numeričkih nizova o kojima smo do sada isključivo i govorili ne postoji ograničenje broja dimenzija.

```
10 REM JEDNODIMENZIONI NIZ
20 DIM B(4)
30 FOR A=1 TO 4
40 LET B(A)=INT (RND*9)+1
50 NEXT A
60 FOR A=1 TO 4
70 PRINT TAB 6;"B(";A;") JE ";
B(A)
80 NEXT A
```

```
B(1) JE 1
B(2) JE 7
B(3) JE 7
B(4) JE 5
```

Sl. 6.4 — Program "Jednodimenzioni niz" i rezultati rada programa


```
10 REM DVODIMENZIONI NIZ
20 REM ------------------
30 DIM A(4,4)
40 FOR B=1 TO 4
50 FOR C=1 TO 4
60 LET A(B,C)=INT (RND*9)+1
70 PRINT "A(";B;",";C;") JE ";
A(B,C)
80 NEXT C
90 NEXT B
100 PRINT TAB 16;"1 2 3 4"
105 PRINT
110 FOR B=1 TO 4
120 PRINT TAB 13;B;TAB 15;A(B,1
);" ";A(B,2);" ";A(B,3);" ";A(B,
4)
130 NEXT B
```

```
A(1,1) JE 1
A(1,2) JE 7
A(1,3) JE 7
A(1,4) JE 5
A(2,1) JE 4
A(2,2) JE 8
A(2,3) JE 3
A(2,4) JE 7
A(3,1) JE 6
A(3,2) JE 3
A(3,3) JE 6
A(3,4) JE 9
A(4,1) JE 4
A(4,2) JE 6
A(4,3) JE 2
A(4,4) JE 7

              1 2 3 4

           1  1 7 7 5
           2  4 8 3 7
           3  6 3 6 9
           4  4 6 2 7
```

Sl. 6.5 — Program "Dvodimenzioni niz" i rezultati rada programa

```
10 REM TRODIMENZIONI NIZ
20 REM ------------------
30 DIM A(3,3,3)
40 FOR B=1 TO 3
50 FOR C=1 TO 3
55 FOR D=1 TO 3
60 LET A(B,C,D)=INT (RND*9)+1
70 PRINT "A(";B;",";C;",";D;")
JE ";A(B,C,D)
75 NEXT D
80 NEXT C
90 NEXT B
```

```
A(1,1,1) JE 1
A(1,1,2) JE 7
A(1,1,3) JE 7
A(1,2,1) JE 5
A(1,2,2) JE 4
A(1,2,3) JE 8
A(1,3,1) JE 3
A(1,3,2) JE 7
A(1,3,3) JE 6
A(2,1,1) JE 3
A(2,1,2) JE 6
A(2,1,3) JE 9
A(2,2,1) JE 4
A(2,2,2) JE 6
A(2,2,3) JE 2
A(2,3,1) JE 7
A(2,3,2) JE 1
A(2,3,3) JE 4
A(3,1,1) JE 4
A(3,1,2) JE 7
A(3,1,3) JE 3
A(3,2,1) JE 9
A(3,2,2) JE 9
```



```
A(3,2,3) JE 3
A(3,3,1) JE 5
A(3,3,2) JE 8
A(3,3,3) JE 2
```

Sl. 6.6 — Program "Trodimenzioni niz" i rezultati rada programa

Korišćenjem osobina nizova uradimo sledeći zadatak. Pretpostavimo da se u nekoj školi koja ima 10 razreda želi ustanoviti koliko u kom razredu ima učenika. Taj rezultat će da prikaže na ekranu sledeći program koji kao ulaz zahteva da se za svakog učenika unosi razred u koji ide, tj. 1—10. Pošto se svaki unos odnosi na jednog učenika, prebrojavanje učenika svešće se na prebrojavanje ulaznih operacija:

```
 20 DIM A(10)
 30 INPUT "UNETI RAZRED/99=KRAJ/";RAZ
 40 IF RAZ=99 THEN GO TO 80
 50 IF RAZ < 1 OR RAZ > 10 THEN GO TO 30
 60 LET A(RAZ)=A(RAZ)+1
 70 GO TO 30
 80 FOR I=1 TO 10
 90 PRINT I;" "; A(I)
100 NEXT I
```

Vidimo da smo definisali jednodimenzioni niz A koji ima deset elemenata. Element A(1) treba da sadrži broj učenika prvog razreda, A(2) broj učenika drugog razreda, . . . , A(10) desetog. Na početku programa naredba DIM vrši anuliranje elemenata matrice A. Naredbom INPUT se za svakog učenika unosi po jedna vrednost koja određuje razred u koji učenik ide. Ako je učenik drugog razreda, sadržaj promenljive RAZ biće 2, ako je učenik šestog razreda, sadržaj promenljive RAZ biće 6, itd. Naredba 60 je bitna za program i važna za razumevanje opravdanosti rada sa nizovima.

LET A(RAZ)=A(RAZ)+1

tumačimo kao povećaj za jedan sadržaj promenljive A čiji indeks predstavlja razred učenika za koga je izvršen upis. Posle toga petljom FOR . . . NEXT prikazujemo broj učenika za svaki razred. Primetite da se na petlju dolazi tek kad se umesto vrednosti za razred izvrši unos vrednosti 99, odnosno pošto smo uneli podatke za sve učenike.

Pogledajte dobro šta bi se desilo kada se ne bi koristila promenljiva tipa niza:

(I) morali bismo da definišemo deset raznoimenih skalarnih promenljivih i da ih na početku programa anuliramo,

(II) deset puta bismo konsultovali sadržaj promenljive RAZ i zavisno od poređenja koristili jednu od 10 operacija sabiranja,

(III) kod kraja programa bilo bi potrebno 10 PRINT operacija. Napišite takav program.

Drugi tip rada sa promenljivim koje imaju strukturu niza odnosi se na one koje imaju nenumeričke sadržaje. Ove nizove možemo smatrati za nizove koji imaju uvek jednu dodatnu dimenziju koja definiše broj znakova koje ima svaki elemenat niza.

Ako naredbom DIM definišemo niz A$ kao:

DIM A$(3,5)

definisali smo tri promenljive A$(1), A$(2) i A$(3), gde svaka od njih ima pet znakova. Pretpostavimo da ima već formiran sledeći sadržaj:

```
P E T A R
M A R K O
T A S A
```

Sadržaj prve promenljive A$(1) je PETAR, A$(2) je MARKO, a promenljive A$(3) je TASA. Kada radimo sa promenljivim koje su nenumeričke, obratite pažnju da:

(I) kada bismo prenosili sadržaj "RADOVAN" u element A(1), prenos bi bio izvršen tako da bi se prenelo samo prvih pet znakova, tj. sadržaj bi bio "RADOV"

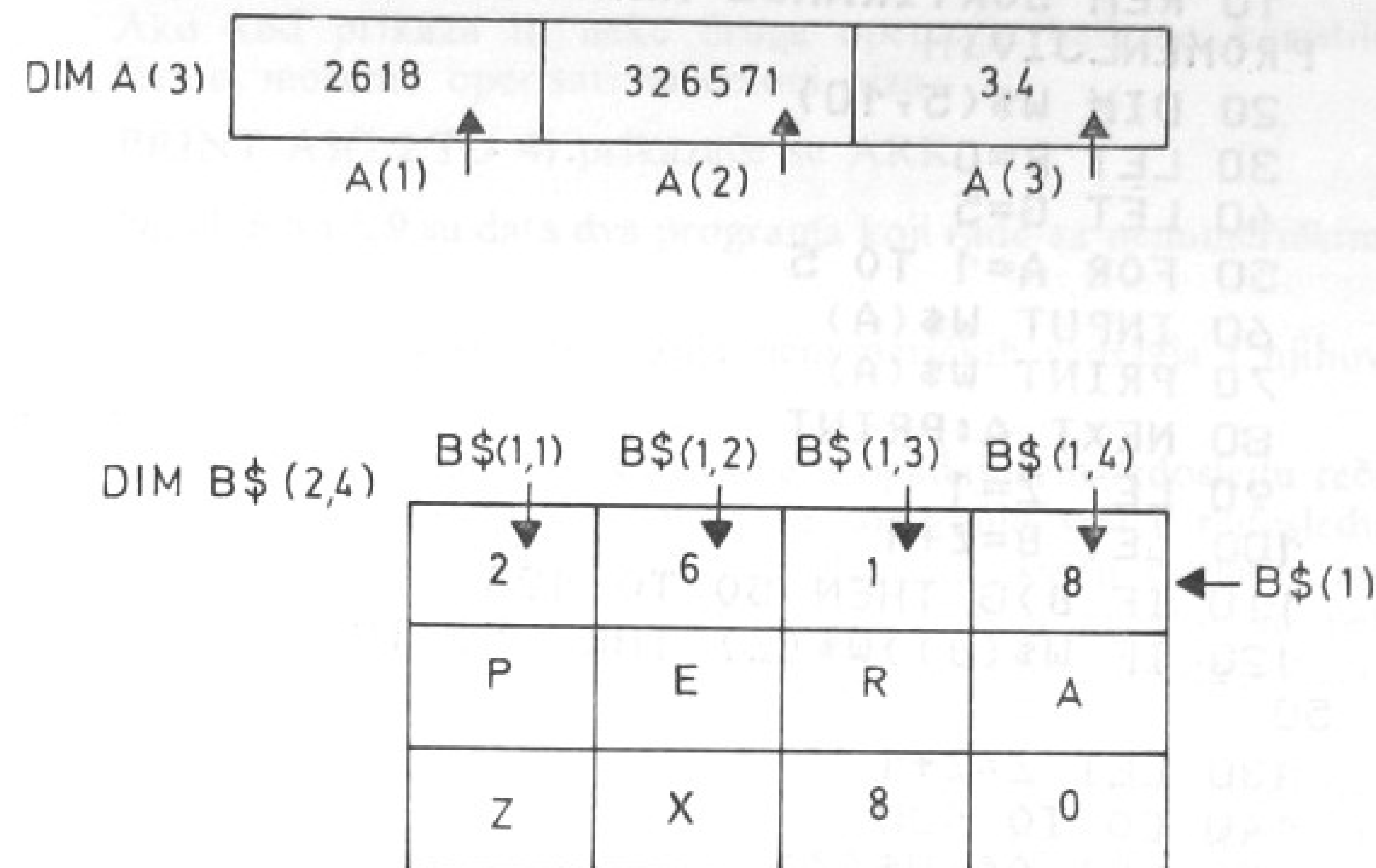
(II) kada bismo prenosili sadržaj "ACO", posle izvršenog prenosa dobili bismo sadržaj "ACO ".

Definisani niz možemo takođe posmatrati i kao dvodimenzioni niz sa elementima:

A$(1,1);A$(1,2);A$(1,3);A$(1,4);A$(1,5)
A$(2,1);A$(2,2);A$(2,3);A$(2,4);A$(2,5)
A$(3,1);A$(3,2);A$(3,3);A$(3,4);A$(3,5)

gde svaki od navedenih elemenata predstavlja sada jedan znak. Tako je A$(1,1)=P; A$(1,2)=E . . . A$(3,5)=blanko. Shodno tome naredba:

PRINT A$(2) prikazaće na ekranu MARKO, a
PRINT A$(3,2) prikazaće na ekranu A.



Sl. 6.7 — Numerički i nenumerički niz

```
10 REM NENUMERICKE PROMENLJIVE
20 DIM A$(4,10)
30 FOR B=1 TO 4
40 INPUT A$(B)
50 NEXT B
60 FOR B=1 TO 4
70 PRINT "A$(";B;") JE ";A$(B)
80 NEXT B
```

```
A$(1) JE ZORAN
A$(2) JE MAJA
A$(3) JE KOMPJUTER
A$(4) JE TASTATURA
```

Sl. 6.8 — Program "Nenumeričke promenljive" i rezultati rada programa

```
10 REM SORTIRANJE NENUMERICKIH
PROMENLJIVIH
20 DIM W$(5,10)
30 LET B=0
40 LET G=5
50 FOR A=1 TO 5
60 INPUT W$(A)
70 PRINT W$(A)
80 NEXT A:PRINT
90 LET Z=1
100 LET B=Z+1
110 IF B>G THEN GO TO 190
120 IF W$(B)>W$(Z) THEN GO TO 1
50
130 LET Z=Z+1
140 GO TO 100
150 LET Q$=W$(Z)
160 LET W$(Z)=W$(B)
170 LET W$(B)=Q$
180 GO TO 130
190 PRINT W$(G)
200 LET G=G-1
210 IF G>0 THEN GO TO 90


NOVI SAD
NIS
BEOGRAD
ZAGREB
ZEMUN

BEOGRAD
NIS
NOVI SAD
ZAGREB
ZEMUN
```

Sl. 6.9 — Program "Sortiranje nenumeričkih promenljivih" i rezultati rada programa

Ako kod prikaza ili neke druge operacije budemo koristili TO formu, možemo operisati sa delom niza.

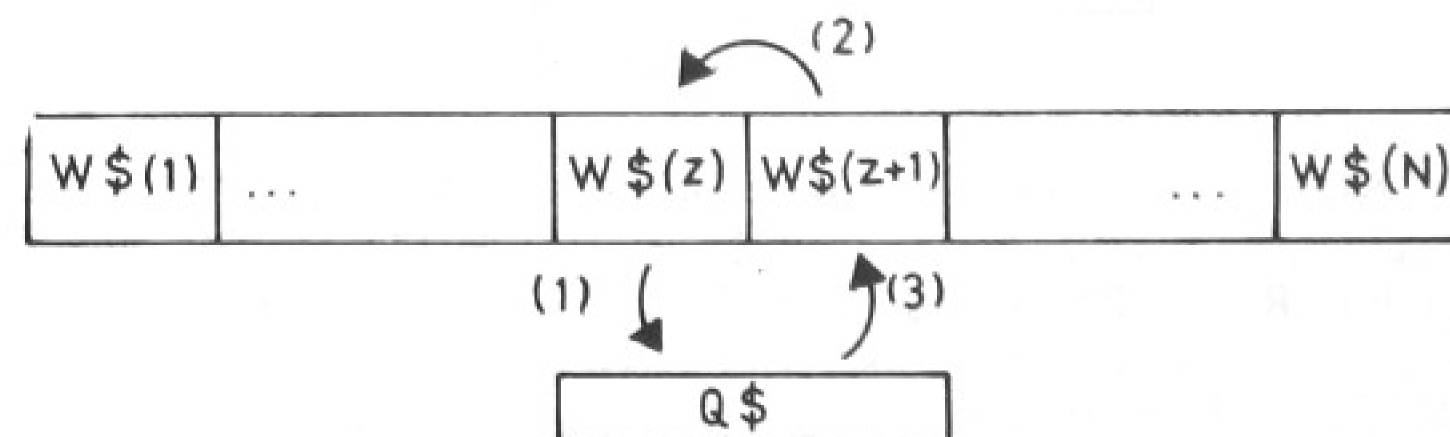
PRINT A$(2,2 TO 4) prikazaće se ARK.

Na sl. 6.8 i 6.9 su data dva programa koji rade sa nenumeričkim nizovima.

Prvi program vrši prihvatanja nenumeričkih sadržaja i njihov prikaz na ekranu.

Sledeći program vrši sortiranje u opadajućem redosledu reči koje se unose. Obratite pažnju da se sortiranje vrši u redosledu koji je unapred određen redosledom slova u okviru engleske latinice. Naredbom DIM W$(5,10) je rezervisan prostor za pet reči koje mogu imati najviše po 10 znakova. Ako želite da to prilagodite svojim potrebama — izmenite program.

Program se sastoji iz većeg broja uzastopnih poređenja sadržaja dve susedne promenljive W$(B) i W$(Z), pri čemu je B=Z+1. Ako je sadržaj promenljive W$(Z) ⩾ W$(B), te dve promenljive su među sobom uređene u opadajućem redosledu. Ako je sadržaj promenljivih W$(Z) < W$(B), tada te dve promenljive treba da zamene mesta. Zamena mesta se vrši uz pomoć promenljive Q$.



Sl. 6.10 — Zamena sadržaja promenljivih W$(Z) i W$(Z+1)

Po završetku rada programa elementi niza W$ su sortirani u opadajućem redosledu. Za razliku od toga, na ekranu ćemo dobiti elemente niza sortirane u rastućem redosledu. To je posledica prikaza elemenata niza W$ u samom procesu sortiranja. U toku procesa elementi niza koji imaju najmanju vrednost premeštaju se na kraj niza i potom se prikazuju.

6.3. *UKRŠTENE REČI*

Gotovo je sigurno da će vas ukrštene reči pcdsetiti na dvodimenzionu promenljivu sa strukturom niza. I na jednoj i na drugoj dimenziji promenljiva ima po pet elemenata. Odlučimo se odmah da prvu dimenziju predstavljaju vrste (u ukrštenim rečima vodoravno), a drugu dimenziju kolone (u ukrštenim rečima uspravno).

1 P	7 E	9 T	A	11 R
2 O	H	O	■	3 O
S	■	4 R	10 O	M
5 L	8 I	B	R	A
6 E	V	A	■	N

Sl. 6.11 — Ukrštene reči

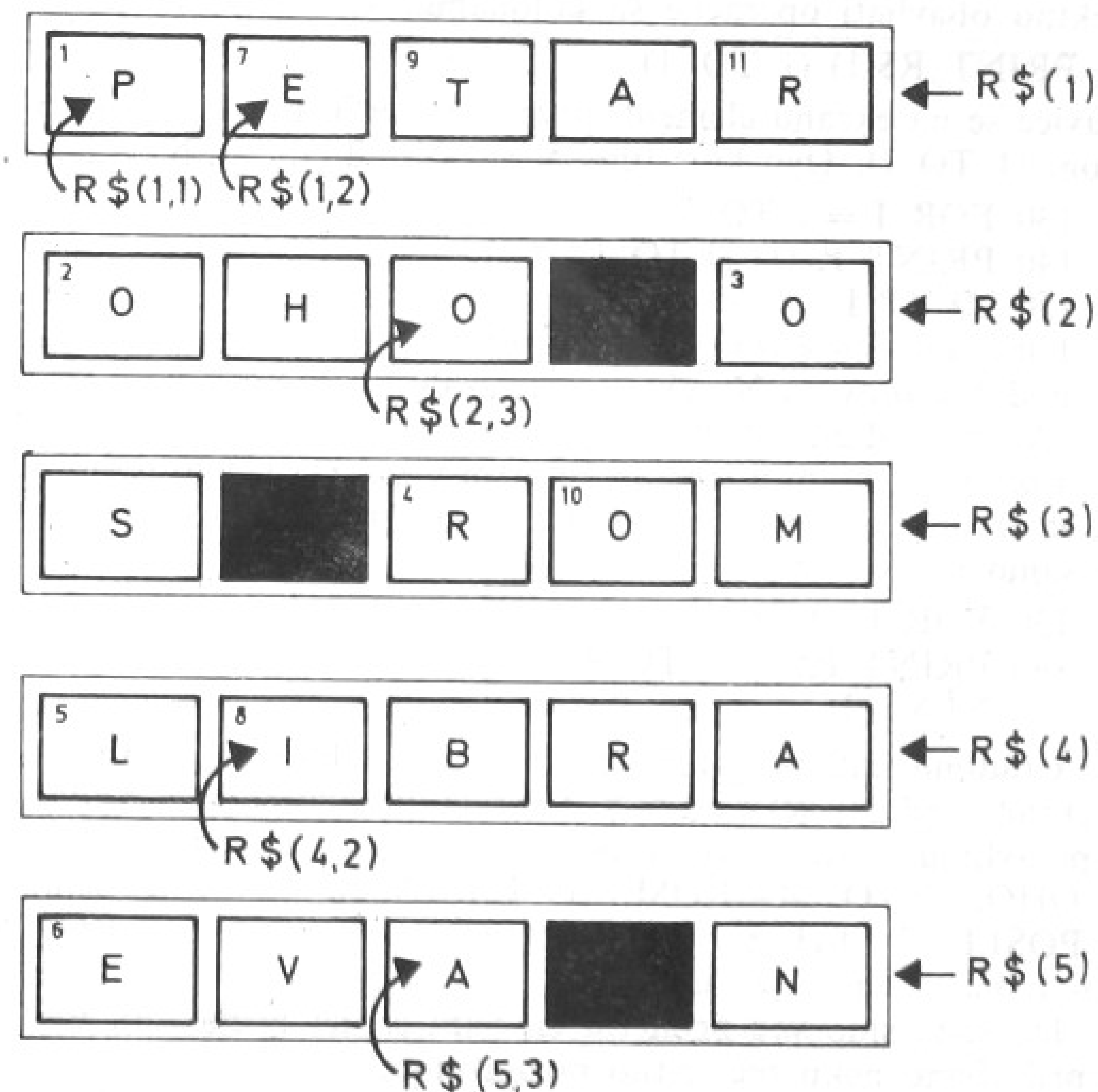
Niz čiji će sadržaj biti ukrštene reči zvaćemo R$. Pošto je u pitanju nenumerički niz koji se definiše sa DIM R$(5,5), vrste će mu biti R$(1), R$(2), R$(3), R$(4) i R$(5). Drugi nivo elemenata ovoga niza je slovo u ukrštenim rečima, kao što je R$(1,1), R$(1,2) . . . Primetite: pošto smo niz definisali tako da se deli na vrste, a zatim na slova u okviru vrste, ne možemo direktno da obavljamo operacije sa kolonama.

Sadržaj niza R$ može se dodeliti direktno naredbom LET:

```
10 DIM R$(5,5)
20 LET R$(1) = "PETAR"
30 LET R$(2) = "OHO■O"
40 LET R$(3) = "S■ROM"
50 LET R$(4) = "LIBRA"
60 LET R$(5) = "EVA■N"
```


ili korišćenjem naredbe READ/DATA. Sada ćemo i prikazati sadržaj niza, odnosno ukrštenih reči na ekranu:

```
 10 DIM R$(5,5)
 20 FOR J=1 TO 5
 30 READ R$(J)
 40 NEXT J
 50 CLS
 60 FOR J=1 TO 5
100 PRINT AT J,5; R$(J)
110 NEXT J
120 DATA "PETAR", OHO■O", "S■ROM", "LIBRA",
    "EVA■N"
```



Sl. 6.12 — Podela ukrštenih reči na vrste i elemente vrsti

U okviru jedne vrste ili kolone može se nalaziti samo jedna reč ili više reči međusobno odvojenih znakom ■.

Ako se u okviru vrste nalazi samo jedna reč, prikazivanje njenog sadržaja svodi se na prikaz sadržaja odgovarajuće vrste. Tako u ovom primeru pod 5 vodoravno nalazi se reč LIBRA, i to u četvrtoj vrsti. Prikazati njen sadržaj možemo sa:

```
PRINT R$(4)
```

Pod četiri vodoravno nalazi se reč ROM, i to u trećoj vrsti. Vidimo da ROM ne zauzima celu vrstu, već samo od treće do pete kolone. Prikaz možemo izvršiti sa:

```
PRINT R$(3) (3 TO 5)
```

Prikaz sadržaja kolona, odnosno odgovora pod uspravno, nešto je složeniji, a to je posledica onoga što smo već rekli: da ne možemo direktno obavljati operacije sa kolonama. Ako napišemo:

```
PRINT R$(1) (1 TO 1)
```

pojaviće se na ekranu element prve vrste R$(1) koji pripada prvoj koloni (1 TO 1). Ovo koristimo u sledećoj FOR . . . NEXT petlji:

```
130 FOR I = 1 TO 5
140 PRINT R (I) (1 TO 1)
150 NEXT I
```

Time smo prikazali sadržaj prve kolone, odnosno ono što piše pod 1 uspravno. Na ekranu ćemo dobiti POSLE.

Ako pogledamo 10 uspravno, vidimo da se tu nalazi reč OR, i to u četvrtoj koloni u trećoj i četvrtoj vrsti. Da bismo dobili samo OR na ekranu, ne treba da prikažemo sadržaj cele kolone na ekranu, već samo njen deo:

```
130 FOR I=3 TO 4
140 PRINT R$(I) (4 TO 4)
150 NEXT I
```

Uradimo sada program koji će na naš zahtev da nam prikazuje na ekranu sadržaj neke od reči. Tih reči ima ukupno 12. Vodoravno se pojavljuju sledeće reči i pod sledećim brojevima: 1 – PETAR, 2 – OHO, 3 – O, 4 – ROM, 5 – LIBRA, 6 – EVA, a uspravno: 1 – POSLE, 7 – EH, 8 – IV, 9 – TORBA, 10 – OR, 11 – ROMAN. Broj 1 ima rešenje i za uspravno i za vodoravno.

Do sada smo već primetili da nam je, da bismo bili u stanju da prikažemo neku reč, bitno poznavanje:

(I) ako je u pitanju vodoravno: broj vrste u kojoj se reč nalazi, kolona u kojoj počinje reč, kolona u kojoj se reč završava;

P(4,1) P(4,2) P(4,3) P(4,4) P(4,5)

1	1	1	1	5
1	2	2	1	3
1	3	2	5	5
1	4	3	3	5
1	5	4	1	5
1	6	5	1	3
2	1	1	1	5
2	7	2	1	2
2	8	2	4	5
2	9	3	1	5
2	10	4	3	4
2	11	5	1	5

Sl. 6.13 — Sadržaj niza P

(II) ako je u pitanju uspravno: broj kolone u kojoj se reč nalazi, vrsta u kojoj reč počinje i vrsta u kojoj se reč završava.

Ove podatke moramo imati za svaku od 12 reči iz ukrštenice. Da bismo unificirali postupke u programu, sve ove podatke stavi-

ćemo u jedan niz koji ćemo zvati P. Ovaj niz će takođe biti dvodimenzioni. Po jednoj dimenziji imaće 12 elemenata, odnosno onoliko koliko u ukrštenim rečima ima reči. Po drugoj dimenziji imaće pet elemenata sledećeg značenja:

— prvi P(I,1): vodoravno (1) ili uspravno (2),

— drugi P(I,2): broj reči (1—11),

— treći P(I,3): ako je vodoravno — vrsta gde se reč nalazi, inače — kolona gde se reč nalazi,

— četvrti P (I,4): ako je vodoravno — kolona gde počinje reč, inače — vrsta gde počinje reč,

— peti P(I,5): ako je vodoravno — kolona gde se završava reč, inače — vrsta gde se završava reč.

Tako će za 1 vodoravno sadržaj ovih elemenata biti:

P (1,1)=1, P(1,2)=1, P(1,3)=1, P(1,4)=1, P(1,5)=5.

Sadržaj nizu P dodaćemo programskom petljom korišćenjem naredbi READ/DATA. Niz P je numerički dvodimenzioni, pa će se naredba READ obavljati u dvostrukoj FOR . . . NEXT petlji.

Dodeljivanje početnih sadržaja vrši se naredbama 30—150. Potom se na ekranu prikazuju ukrštene reči.

Naredbe programa 210 — 260 služe za pojašnjenje i sprovođenje operacije ulaza. Istovremeno se vrše i kontrole unetih vrednosti.

U nastavku programa određuje se kome elementu niza P pripada reč čiji se prikaz zahteva. Ovde se takođe sprovodi još jedna kontrola. Tako, pri unosu, može se reći da se traži vodoravno pod brojem 8. U tom slučaju promenljiva VU = 1, a BR = 8. Ako pogledate u ukrštene reči, tako definisana reč ne postoji.

Ovu kontrolu nismo mogli sprovesti odmah posle ulaza zato što ona predstavlja ustanovljavanje važećih veza između pojmova vodoravno, uspravno i pod kojim brojem, pa se može sprovesti samo pretraživanjem niza P.

Pretraživanje niza P vrši se naredbama 270—300. U slučaju da željena reč postoji u ukrštenici — program nastavlja sa radom, inače — ide se na naredbu 210:

```
10 REM UKRSTENE RECI
20 REM DODELA SADRZAJA NIZOVIMA R$ i P
30 DIM R$(5,5)
40 FOR J=1 TO 5
50 READ R$(J)
60 NEXT J
```



```
70 DATA "PETAR","OHO■O","S■ROM", "LIBRA",
   "EVA■N"
75 DIM P(12,5)
80 FOR J=1 TO 12
90 FOR K=1 TO 5
100 READ P(J,K)
110 NEXT K
120 NEXT J
130 DATA 1,1,1,1,5,1,2,2,1,3,1,3,2,5,5,
140 DATA 1,4,3,3,5,1,5,4,1,5,1,6,5,1,3
150 DATA 2,1,1,1,5,2,7,2,1,2,2,8,2,4,5
160 DATA 2,9,3,1,5,2,10,4,3,4,2,11,5,1,5
170 CLS
180 FOR J=1 TO 5
190 PRINT AT J+5, 10;R$(J)
200 NEXT J
210 PRINT' "KOJU REC ZELITE DA PRIKAZETE?"
220 INPUT "VODORAVNO ILI USPRAVNO 1 ili 2"; VU
230 IF NOT (VU=1 OR VU=2) THEN BEEP .5,1: GO TO 220
240 INPUT "POD KOJIM BROJEM 1 do 11"; BR
250 LET BR = INT BR
260 IF NOT (BR > 0 AND BR < 12) THEN BEEP.5,1:
    GO TO 240
265 REM PRETRAZIVANJE NIZA P
270 LET IND = 0
280 FOR J = 1 TO 12
290 IF P(J,1)=VU AND P(J,2)=BR THEN LET IND=J
300 NEXT J
310 IF IND > 0 THEN GO TO 380
320 BEEP .5,1
330 IF VU=1 THEN LET A$ = "VODORAVNO"
340 IF VU=2 THEN LET A$ = "USPRAVNO"
350 PRINT "NE POSTOJI REC "; A$
360 PRINT "POD BROJEM"; BR
370 GO TO 210
380 IF VU=2 THEN GO TO 410
390 PRINT R$(P(IND,3)) (P(IND,4) TO P(IND,5))
395 PAUSE 150
400 GO TO 170
410 FOR J = P (IND,4) TO P(IND,5)
```

```
420 PRINT R$(J) (P(IND,3) TO P(IND,3)):NEXT J
425 PAUSE 150
430 GO TO 170
```

Posle se prikazuje tražena reč na jedan od dva opisana načina, zavisno od toga da li je tražena reč pod vodoravno ili uspravno.

Pogledajmo sve to još jednom na primeru reči ROM, 4 vodoravno. Taj zahtev se definiše unošenjem 1—vodoravno, 4—pod kojim brojem. Posle operacije unosa sadržaj promenljive VU = 1, a promenljive BR = 4.

Ovoj kombinaciji vrednosti VU i BR odgovara četvrta vrsta P, što znači da će sadržaj promenljive J postati 4.

Pošto je tražena reč bila pod vodoravno, prikaz će se vršiti naredbom 390. U ovoj naredbi se pojavljuju:

— P(IND,3) = P(4,3) = 3, tj. reč se nalazi u trećoj vrsti;
— P(IND,4) = P(4,4) = 3, tj. reč počinje od treće kolone;
— P(IND,4) = P(4,4) = 5, tj. reč se završava u petoj koloni.

Naredba PRINT je za ove konkretne vrednosti:

PRINT R$(3) (3 TO 5)

6.4. VEŽBE

1) Definisati jedan jednodimenzioni numerički niz koji sadrži 20 elemenata. Potom korišćenjem naredbe INPUT dodelite elementima niza posebne brojne vrednosti. Programski utvrdite koja je od njih najveća i tu vrednost prikažite na ekranu.

2) Definišite jedan dvodimenzioni numerički niz koji ima 5×5 elemenata. Korišćenjem naredbe INPUT dodelite elementima niza posebne brojne vrednosti. Programom utvrditi koliko ima članova niza, čija je vrednost veća od 10.

3) Definišite dva jednodimenziona numerička niza A i B koji sadrže po 10 elemenata. Naredbom INPUT dodelite im posebne brojčane vrednosti. Posle toga izračunajte i prikažite vrednosti za D.

$$D = \sqrt{\sum_{i=1}^{10} (A_i - B_i)^2}$$

4) Napravite programe koji vrše formiranje dve matrice A i B i njihov tabelaran prikaz na ekranu. Posle toga napraviti programe koji rade sabiranje, oduzimanje i množenje te dve matrice (vidi prilog 2).

5) Definišite jednodimenzioni numerički niz F koji ima 10 elemenata. Potom programski svim elementima, osim prvog i poslednjeg, promenite sadržaj sa:

$$F_i = \frac{F_{i-1} + F_i + F_{i+1}}{3}$$

i sadržaj izmenjenog niza prikazati na ekranu.

6) Definisati jednodimenzioni numerički niz A koji ima 20 elemenata. Korišćenjem naredbe INPUT elementima niza dodelite posebne brojne vrednosti. Zatim programski izračunati i prikazati vrednost za B.

$$B = \cfrac{1}{1 + \cfrac{a_1}{1 + \cfrac{a_2}{1 + \cfrac{a_3}{\ddots}}}}$$

7) Uredite program koji rešava sledeći sistem linearnih jednačina:

$$\begin{aligned}
a_{11}X_1 &&&&&= b_1 \\
a_{21}X_1 + a_{22}X_2 &&&&&= b_2 \\
a_{31}X_1 + a_{32}X_2 + a_{33}X_3 &&&&&= b_3 \\
a_{41}X_1 + a_{42}X_2 + a_{43}X_3 + a_{44}X_4 &&&&&= b_4 \\
a_{51}X_1 + a_{52}X_2 + a_{53}X_3 + a_{54}X_4 + a_{55}X_5 &&&&&= b_5
\end{aligned}$$

Vrednost za elemente nizova A i B se učitavaju, a vrednost elemenata niza X se računa i prikazuje kao rešenje. A je dvodimenzioni numerički niz sa 5×5 elemenata. B i X su jednodimenzioni numerički nizovi sa 5 elemenata.

8) Prvog januara 1985. godine bio je utorak. Treba, zavisno od zahteva, da prikazujemo na ekranu kalendar sledećeg izgleda:

JANUAR 1985. GODINE

1 UTORAK	2 SREDA	3 CETVRTAK
4 PETAK	5 SUBOTA	6 NEDELJA
7 PONEDELJAK	8 UTORAK	9 SREDA

Podatke o nazivima meseci i dana preuzimajte iz dva nenumerička (STRING) niza.



Boje i grafika

7.1. BOJE

Računar Spectrum raspolaže sa 8 različitih boja. Boje su numerisane ciframa od 0 — 7 na sledeći način:

0 — crna
1 — plava
2 — crvena
3 — ljubičasta
4 — zelena
5 — plavo-zelena (Cyan)
6 — žuta
7 — bela

U slučaju da je TV prijemnik priključen na Spectrum izveden u crno-beloj tehnici, umesto boja se raspolaže sa raznim varijantama sive boje između crne (0) i bele (7).

Pomoću naredbi koje upravljaju bojom mogu se kontrolisati:

— boja središnjeg dela ekrana — to se čini naredbom PAPER,
— boja ivica ekrana — to se čini naredbom BORDER,
— boja zapisa — to se čini naredbom INK.

Ako želimo pisati po žutoj osnovi ekrana crvenim slovima, a da pri tom boja ivice ekrana bude crna, biće:

```
PAPER 6
INK 2
BORDER 0
```


Ukoliko se zahtev za bojama ne specificira, podrazumeva se PAPER i BORDER jednak 7, a INK jednak nuli. To je pisanje crnim mastilom po beloj podlozi.

Uz INK i PAPER kao broj boje može se naći i 8 i 9.Oni ne odgovaraju direktno bojama, već imaju specijalna značenja: 8 ima značenje iste boje, a 9 znači boju kontrasta.

Bela boja je kontrast za tamne (crna — ljubičasta), a crna boja je kontrast za svetle (zelena — bela).

Boje definisane sa PAPER i INK mogu se koristiti na dva načina. Prvi je tzv. globalni način upotrebe. Ako date naredbu PAPER 6 i posle toga CLS, boja središnjeg dela ekrana postaće žuta. Takođe, ako date naredbu INK 2, slova će biti upisana crvenom bojom. Takav odnos "mastilo-papir" važiće sve dok se ponovo ne pojave naredbe PAPER i INK u globalnoj formi.

Lokalna upotreba boje definisana je uz naredbu PRINT. Ako damo naredbu PRINT INK 1; PAPER 7; "DOBAR DAN", prikazaće se tekst DOBAR DAN plavim slovima na beloj podlozi. Ako damo ponovo naredbu PRINT "DOBAR DAN", tekst i podloga biće iste boje koja je već ranije definisana kao globalna.

Lokalna upotreba važi takođe i za naredbu INPUT. Probajte: INPUT (INK 2; PAPER 6; "KOJA JE OVO BOJA"); A.

Sledeći program koristi FOR ... NEXT petlje da bi se prikazale sve moguće kombinacije za BORDER, PAPER i INK:

```
 10 REM BOJE
 20 FOR B = 0 TO 7
 30 FOR P = 0 TO 7
 40 FOR I = 0 TO 7
 50 BORDER B
 60 PAPER P : CLS
 70 INK I
 80 PRINT AT 10,5;"BORDER";B;
          AT 11,5;"PAPER";P;
          AT 12,5;"INK";I
 90 PAUSE 60
100 NEXT I
110 NEXT P
120 NEXT B
```

Sledeći program crta piramidu sastavljenu od malih blokova čija se boja menja na slučajan način:

```
 10 REM PIRAMIDA
 20 BORDER 7
```

```
30 CLS
40 LET B=16
50 LET T=0
60 LET S=0
70 LET L=20
80 LET T=T+B
90 FOR N=S TO S+B*2—2
100 PRINT AT L,N; INK INT (RND*
    6)+1; CHR$ (143)
110 BORDER INT (RND*6)+1
120 NEXT N
130 LET L=L—1
140 LET B=B—1
150 LET S=S+1
160 IF B> 0 THEN GO TO 80
170 BORDER 1
180 GO TO 180
```

Linija 180 GO TO 180 predstavlja "petlju" koja se zatvara samo na istom broju naredbe gde i počinje, pa prema tome ne vrši nikakvu funkciju. U ovome programu služi da po stvarnom završetku rada programa odloži izlaz poruke, da je program završio sa radom — kako se ne bi pokvarila slika piramide. Program se prekida naredbom BREAK.

7.2. NAREDBE PLOT, DRAW, CIRCLE

Korišćenjem naredbe PLOT x,y mogu se markirati, tj. obojiti elementi ekrana koji se nalaze na pozicijama x,y. Argumenti uz naredbu PLOT odnose se na osnovne ćelije pozicija na ekranu (engl. Pixel). Koordinata x ide od krajnje leve pozicije ekrana udesno i može imati vrednosti 0—255(256=32*8). Koordinata y ide od dna ekrana nagore i može imati vrednosti 0 - 175(176=22*8). Prema tome, krajnje pozicije ekrana su: (0,0) leva donja, (0,175) leva gornja, (255,0) desna donja i (255,175) desna gornja. Primetite da je uz naredbu PLOT koordinatni sistem orijentisan nagore – za razliku od funkcije AT u naredbi PRINT. Takođe treba imati na umu da je uz PLOT redosled argumenata x, y, a kod PRINT AT obrnuto.

Sledeći program korišćenjem jedne FOR . . . NEXT petlje u okviru koje se nalazi naredba PLOT crta funkciju y=x:

```
10 FOR I=0 TO 175
20 PLOT I,I
30 NEXT I
```


Koordinatni početak se nalazi u levom donjem uglu, pa je ekran televizora deo I kvadranta. Ako želimo da obavljamo prikaze tačaka koje imaju negativne koordinate, moraćemo da "pomerimo" koordinatni sistem.

Postavimo centar koordinatnog sistema u tačku (127,80) i nacrtajmo zatim grafik funkcije y=sinx za vrednosti argumenta (-2π, 2π):

```
10 PLOT 127,80
20 FOR I=0 TO 255
30 PLOT I, 80+60*SIN((I/64—2)*Pi)
40 NEXT I
```

Prvo se prikaže "novi" koordinatni početak tačka (127,80). Argument x naredbe PLOT je upravljački promenljiva FOR . .NEXT petlje I koja uzima vrednosti 0—255. Za te vrednosti ugao sinusne funkcije uzima vrednosti od —2PI do +1.984PI.

Argument J u naredbi PLOT uzima vrednosti 20—140. Broj 80 koji se javlja u opisu argumenta y održava pomeranje koordinatnog sistema na poziciju 80.

Proširimo prethodni zadatak tako što ćemo nacrtati i koordinatni sistem:

```
50 FOR I=0 TO 150
60 PLOT 127, I
70 LET J=INT ((I—80)/60)
80 IF J=(I—80)/60 THEN
   PRINT AT 21—INT(I/8),16; J
90 NEXT I
```

Prethodni deo programa crta y-osu koordinatnog sistema. X-osa se može opisati na sledeći način:

```
100 FOR I=0 TO 255
110 PLOT I,80
120 LET J=INT(I/64)
130 IF J= (I/64) THEN
    PRINT AT 11, INT (I/8); (I/64—2); "*PI"
140 NEXT I
```

Koordinatni sistem mogli smo postaviti i naredbom DRAW. Ova naredba kao argumente takođe ima koordinate x,y, ali one sada predstavljaju relativni pomeraj u odnosu na poslednju tačku definisanu nekom PLOT, DRAW ili CIRCLE naredbom. Tako, ako kažemo:

```
PLOT 100, 100
```

misli se da želimo da markiramo neku tačku koja se nalazi na poziciji 100, 100. Ako potom kažemo:

```
DRAW 20, —10
```

izdajemo naredbu da se povuče prava linija između tačke sa koordinatama (100, 100) i tačke koja za 20 ima veću apscisu i za 10 manju ordinatu. Znači, želimo spojiti pravom linijom tačke (100,100) i (120,90).

Sledeći program povlači prave linije između tačke sa fiksnim koordinatama (128,86) i druge tačke odabrane na slučajan način:

```
10 REM NAREDBA DRAW
20 PAPER 7
30 LET A=INT (RND*8)
40 LET B=INT (RND*7)
50 IF A=B THEN GO TO 40
60 BORDER A
70 INK B
80 CLS
90 LET C=INT (RND*256)—128
100 LET D=INT (RND*172)—85
110 PLOT 128,86
120 DRAW C, D
130 BEEP .1, RND*100—50
140 IF RND>0.1 THEN GO TO 90
150 RUN
```

Naredbom CIRCLE možemo na ekranu prikazivati krugove. Argumenti koji idu uz ovu naredbu su x, y, r, tj. koordinate centra kruga i dužina poluprečnika. Sledeći program prikazuje familiju krugova čiji se centar i dužina poluprečnika određuju na slučajan način:

```
10 REM NAREDBA CIRCLE
20 BORDER 5 : PAPER 7 : CLS
30 CIRCLE INK RND*6; 128+RND*10—RND*10,
   86+RND*7—RND—7, RND*65
40 IF RND>.92 THEN CLS
50 BEEP RND/3, RND*100—30
60 GO TO 30
```

Svako novo pozivanje na funkciju RND generiše novi slučajan broj. Tako izraz RND—RND ne mora da bude jednak nuli.

Sledeći program takođe prikazuje familiju krugova. Pozicija njihovih centara se određuje na slučajan način. Za svaku poziciju

centra prikazuje se nekoliko krugova čiji se poluprečnici međusobno razlikuju za 1, pa se tako stiče utisak popunjenog kruga:

```
10 REM NAREDBA CIRCLE
20 LET A=INT (RND*10)
30 IF A=7 THEN GO TO 20
40 INK A
50 LET X=30 + INT (RND*190)
60 LET Y=30 + INT (RND*100)
70 FOR N=10 TO INT (RND*25)
80 CIRCLE X,Y,N
90 NEXT N
100 GO TO 20
```

7.3. DVODIMENZIONE TRANSFORMACIJE

Osnovne dvodimenzionalne transformacije koje se mogu vršiti sa grafičkim likovima su: translacija, rotacija i promena veličine grafičkog lika. Dve (ili više) transformacija mogu se kombinovati ili povezivati u jednu koja ima isti efekat kao sekvenca prvobitne dve transformacije. Tako, ako sa A označimo transformaciju tipa translacije, a sa B transformaciju tipa promene veličine lika, tada složena transformacija C=AB mora dati isti efekat kao sprovođenje prve, pa druge transformacije.

Svaka od navedenih transformacija, polazeći od početnih koordinata neke tačke (X,Y), formira nove koordinate te tačke (X,′ Y′). Ako se u originalnoj definiciji grafički lik sastoji od pravih linija, dovoljno je sprovesti postupak transformacije samo na presečne tačke prvih. Na jednom takvom jednostavnom primeru opisaćemo navedene tipove transformacija.

Translacija se izvodi sa:

$$x' = x + Tx \qquad y' = y + Ty$$

Posmatrajmo jedan prosti lik — trougao definisan sa tri temena (20,10), (40,20), (30,50). Tx=20, Ty=30 znači translaciju 20 pozicija udesno i 30 pozicija nagore. Nova temena su (40,40), (60,50), (50,80).

Rotacija. Da bismo rotirali neku tačku (X,Y) za ugao θ, u pravcu kazaljke na satu oko koordinatnog početka potrebno je izvršiti sledeću transformaciju:

$$x = x \cos \theta + y \sin \theta \qquad y' = -x \sin \theta + y \cos \theta$$

Ako je ugao rotacije $\theta=20°$, tada je $\sin\theta = 0,34202$, a $\cos\theta = 0,93969$.

X	Y	X cos θ	X sin θ	Y sin θ	Y cos θ	X′	Y′
20	10	18.39	6.84	3.42	9.39	21.81	2.55
40	20	37.58	13.68	6.84	18.39	44.42	4.71
30	50	28.19	10.26	17.10	46.98	45.29	36.72

Nove koordinate x′, y′ temena trougla su (21.81, 2.55), (44.42, 4.71), (45,29, 36.72).

Promena veličine lika. Ova transformacija se izvodi sa:

$x' = xS_x \quad y' = yS_y$

Ako uzmemo $S_x = S_y = 0.5$, znači da želimo da vršimo umanjenje lika za dva puta. Nove koordinate temena trougla biće (10, 5), (20, 10), (15, 25). Postupak promene veličine lika je takođe vezan za poziciju početka koordinatnog sistema.

Ako su S_x i S_y veći od jedinice, lik se uvećava. Ako S_x i S_y nisu jednaki, tada se lik deformiše tako što se izdužuje ili sužava. Ako se koriste negativne vrednosti za S_x, S_y, dobija se efekat slike u ogledalu.

Sledeći programi vrše translaciju trougla. Promenljive AX, AY, BX, BY, CX, CY su koordinate temena trougla A,B, i C. Programska rutina 200 vrši prikaz trougla:

```
10 REM TRANSLACIJA TROUGLA
20 REM KOORDINATNI SISTEM
30 PLOT 0,0 : DRAW 250,0
40 PLOT 0,0 : DRAW 0,170
50 LET AX=40 : LET AY=40
60 LET BX=60 : LET BY=120
70 LET CX=90 : LET CY=70
80 GO SUB 200
90 REM TRANSLACIJA
100 LET TX=20
110 LET TY=30
120 LET AX=AX+TX : LET AY=AY+TY
130 LET BX=BX+TX : LET BY=BY+TY
140 LET CX=CX+TX : LET CY=CY+TY
145 INK 4
```



```
150 GO SUB 200
155 INK 9
160 STOP
190 REM PRIKAZ TROUGLA
200 PLOT AX, AY
210 DRAW BX-AX, BY-AY
220 DRAW CX-BX, CY-BY
230 DRAW AX-CX, AY-CY
240 RETURN
```

U prethodnom programu sprovedite sledeće izmene da bi program sprovodio promenu veličine lika trougla:

```
10 REM PROMENA VELICINE LIKA TROUGLA
90 REM PROMENA VELICINE LIKA
100 LET SX=0.5
110 LET SY=0.7
120 LET AX=AX*SX : LET AY=AY*SY
130 LET BX=BX*SX : LET BY=BY*SY
140 LET CX=CX*SX : LET CY=CY*SX
```

U izmenama programa koje treba sprovesti da bi program obavljao rotaciju uvedena je nova promenljiva P. Ona se uvodi da se sačuva vrednost X potrebna za računanje Y′. Za razliku od translacije i promene veličine, ovde je $x'=f(x,y)$ i $y'=f(x,y)$:

```
10 REM ROTACIJA TROUGLA
90 REM ROTACIJA
100 LET TETA=30
110 LET TETA=TETA*PI/180
120 LET P=AX
125 LET AX=AX*COS(TETA)+AY*SIN(TETA):
    LET AY=-P*SIN(TETA)+AY*COS(TETA)
130 LET P=BX
135 LET BX=BX*COS(TETA)+BY*SIN(TETA):
    LET BY=-P*SIN(TETA)+BY*COS(TETA)
140 LET P=CX
145 LET CX=CX*COS(TETA)+CY*SIN(TETA):
    LET CY=-P*SIN(TETA)+CY*COS(TETA)
```

Povezivanje transformacija. Veoma retko se transformacija lika obavlja samo jednom osnovnom transformacijom. Videli smo da se osnovne transformacije odnose na koordinatni početak. Ako bismo

trougao sa kojim smo radili osnovne transformacije želeli da rotiramo oko neke druge tačke, taj postupak bismo izveli povezivanjem translacije, rotacije i translacije.

Povezivanjem transformacija ne sme biti pokvaren redosled izvršavanja osnovnih transformacija. Ako prvo izvršimo translaciju trougla sa, na primer, $T_x=-80$, $T_y=0$, a potom rotaciju za 90°, nećemo dobiti sliku trougla na istoj poziciji kao u slučaju da smo te dve operacije izvršili u obrnutom redosledu.

Najveća korist povezivanja transformacija je mogućnost predstavljanja sekvence transformacija jednom transformacijom. Na primer — sledeća sekvenca:

$x' = y \qquad y' = -x$

i

$x'' = x'-80 \qquad y'' = y'$

može se povezivanjem svesti samo na:

$x'' = y - 80 \qquad y'' = -x$

Korišćenje povezivanja ima i tu prednost što se čitava sekvenca povezivanja može kompaktno izraziti. To izražavanje se dosta pojednostavljuje ako se koriste osobine matrica da bi se definisale transformacije.

Predstavljanje osnovnih i povezanih transformacija pomoću matrica. Dvodimenziona transformacija se može predstaviti korišćenjem 3×3 matrice. Transformacija tačke (x, y) u novu tačku (x′, y′) bilo kojom sekvencom translacija, rotacija i promena veličine lika može se predstaviti kao:

$$[x' \ y' \ 1] = [x \ y \ 1] \begin{bmatrix} a & d & 0 \\ b & e & 0 \\ c & f & 1 \end{bmatrix}$$

Upotrebljena 3×3 matrica može se imenovati, čime se dosta pojednostavljuje opisivanje celoga postupka transformacije. Postojanje jedinice uz opis tačaka (x, y) i (x′, y′) omogućuje vršenje transformacija 3×3 matricom na isti način u svim slučajevima osnovnih transformacija.

Osnovne transformacije se preko matrica predstavljaju na sledeći način:

Translacija:

$$[x' \ y' \ 1] = [x \ y \ 1] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{bmatrix}$$

ili predstavljeno na drugi način:

$x' = y + T_x$

$y' = y + T_y$

$1 = 1$

Prve dve jednakosti su već opisane, a poslednja proizilazi iz postojanja jedinice u opisu tačaka (x, y) i (x′ y′) koja omogućuje opis transformacije na isti način u sva tri osnovna slučaja.

Rotacija:

$$[x' \ y' \ 1] = [x \ y \ 1] \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Promena veličine lika:

$$[x' \ y' \ 1] = [x \ y \ 1] \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Pri izvršavanju postupka povezivanja transformacija, operacije koje treba da se obave svode se na operacije množenja matrica. Pretpostavimo da želimo da izvršimo promenu veličine lika sa parametrima $S_x=S_y=2$, a potom translaciju sa parametrima $T_x=10$ $T_y=0$. U tom slučaju imamo:

Promena veličine lika:

$$[x' \ y' \ 1] = [x \ y \ 1] \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Translacija:

$$[x'' \ y'' \ 1] = [x' \ y' \ 1] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 10 & 0 & 1 \end{bmatrix}$$

Rezultat, odnosno koordinate (x'', y'') se mogu direktno izraziti u funkciji polazne tačke (x, y) sa:

$$[\,x''\ y''\ 1\,] = [\,x\ y\ 1\,] \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 10 & 0 & 1 \end{bmatrix}$$

Prethodni izraz se može pojednostaviti operacijom množenja dve 3 × 3 matrice koje opisuju osnovne trasformacije:

$$[\,x''\ y''\ 1\,] = [\,x\ y\ 1\,] \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 10 & 0 & 1 \end{bmatrix}$$

Na taj način ponovo smo dobili istu formu kao i kod osnovnih transformacija. Dobijena transformacija se može predstaviti i na drugi način:

$$x'' = x \cdot 2 + 10$$
$$y'' = y \cdot 2$$

Pretpostavimo sada da želimo da odredimo transformaciju koja nam omogućuje da izvršimo rotaciju neke tačke (x, y) u pravcu kretanja kazaljke na satu za ugao θ, a oko tačke sa koordinatama (R_x, R_y). Rotacija, na način kako je opisana, obavlja se samo oko koordinatnog početka. Zato ćemo prvo translacijom postaviti koordinatni početak u tačku (R_x, R_y):

$$[\,x'\ y'\ 1\,] = [\,x\ y\ 1\,] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -R_x & -R_y & 1 \end{bmatrix}$$

Rotacija se opisuje sa:

$$[\,x''\ y''\ 1\,] = [\,x'\ y'\ 1\,] \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Posle toga moramo obaviti ponovo translaciju u suprotnom smeru da bismo "vratili" koordinatni početak na svoje mesto:

$$[\,x'''\ y'''\ 1\,] = [\,x''\ y''\ 1\,] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ R_x & R_y & 1 \end{bmatrix}$$

Navedene tri osnovne operacije mogu se povezati u jednu:

$$[\,x'''\ y'''\ 1\,] = [\,x\ y\ 1\,] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -R_x & -R_y & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ R_x & R_y & 1 \end{bmatrix}$$

Ako su poznate vrednosti za R_x, R_y i θ, može se sprovesti množenje matrica i dobiti jedna matrica transformacije.

Primer dvodimenzione transformacije. Sledeći program obavlja sve tri osnovne transformacije jednog trougla u sledećem redosledu: (a) rotacija, (b) translacija, (c) promena veličine lika. Posle svake osnovne transformacije prikazuje se lik trougla. Zatim se operacijama množenja matrica dobija matrica povezane transformacije i sprovodi povezana transformacija. Prikazuje se trougao sada u crvenoj boji, i to na istom mestu gde je bio posle tri osnovne transformacije.

Osnovne promenljive koje se pojavljuju u programu su promenljive A i T. Obe su trodimenzionalne promenljive sa strukturom niza numeričkog tipa. Niz T sadrži opise transformacija koje treba sprovesti, a niz A koordinate temena trougla.

Svi opisi transformacija koje smo do sada pominjali, mogu se predstaviti dvodimenzionim nizovima, odnosno u matematici se izražavaju preko matrica. Uvođenje treće dimenzije u nizu T ima za cilj uniformno obrađivanje svih transformacija. Na taj način se donekle (u ovom zadatku) prevazilazi problem nemogućnosti definisanja argumenata programske rutine koja se poziva GO SUB naredbom. Pâr naredbi GO SUB i RETURN omogućavaju isključivo prelaz na, i povratak iz programske rutine. Operaciju množenja matrica možemo programirati u okviru programske rutine. Neka A, B, C budu nazivi dvodimenzionih nizova koji odgovaraju matricama koje se množe i rezultatu. Ako u programu želimo da obavimo množenje C=A*B, sve što treba da uradimo jeste da aktiviramo odgovarajuću naredbu GO SUB, a programska rutina će sprovesti množenje.

Problem nastaje kada želimo da obavimo množenje neke druge dve matrice, na primer Z=X*Y. Programsku rutinu za množenje matrica ne možemo direktno da koristimo zato što ona radi sa A, B i C. Ostaje nam, dakle, ili da napišemo drugu programsku rutinu koja radi sa X, Y i Z, ili da pre povezivanja rutine iz-

vršimo prilagođavanje nazivima matrica koji se koriste u rutini. To se može postići, na primer, sledećim operacijama prenosa sadržaja: X→A, Y→B pre, i: C→Z posle rada programske rutine. Ove tri operacije nisu mnogo jednostavnije i ne obavljaju se brže od množenja matrica. Iz tog razloga uveli smo još jednu dimenziju, pa su nizovi A i T trodimenzionalni.

Niz A (I, J, K) određuju koordinate temena trougla. Definiše se naredbom DIM A(4,3,3). Indeksi I, J, K imaju sledeće značenje:

I — položaj temena. I=1 početni položaj, I = 2 položaj posle prve transformacije (rotacije), I = 3 položaj posle druge transformacije (translacije), I = 4 položaj posle treće transformacije (promena veličine).

J — teme trougla. J = 1 teme A, J = 2 teme B, J = 3 teme C.

K — koordinate temena. K = 1 koordinata X, K = 2 koordinata Y, K = 3 konstanta 1.

Niz T(L,M,N) određuje opise transformacija. Definiše se naredbom DIM T(4,3,3). Indeksi L,M,N imaju sledeća značenja:

L — broj transformacione matrice. L = 1 pomoćna transformaciona matrica (koristi se za prijem rezultata množenja dve matrice), L=2 matrica prve transformacije (rotacije), L = 3 matrica druge transformacije (translacije), L = 4 matrica treće transformacije (promena veličine).

M — broj vrste transformacione matrice. M = 1 prva vrsta, M = 2 druga vrsta, M = 3 treća vrsta.

N — broj kolone transformacione matrice. N=1 prva kolona, N = 2 druga kolona, N = 3 treća kolona.

Program je podeljen u tri dela. Prvi obavlja inicijalizacije nizova A i T i vrši prikaz koordinatnog sistema. Drugi deo obavlja pojedinačne transformacije i prikaze trougla posle svake transformacije. Treći deo obavlja poveznu transformaciju i prikaz trougla:

```
10 REM TRANSFORMACIJA TROUGLA
20 REM INICIJALIZACIJE
30 DIM A(4,3,3)
40 DIM T(4,3,3)
50 PLOT 0,0 : DRAW 250,0
60 PLOT 0,0 : DRAW 0,170
70 FOR J=1 TO 3
80 FOR K=1 TO 3
90 READ A(1,J,K)
```



```
100 NEXT K
110 NEXT J
120 DATA 50,0,1,150,0,1,100,30,1
130 FOR L=2 TO 4
140 FOR M=1 TO 3
150 FOR N=1 TO 3
160 READ T(L,M,N)
170 NEXT N
180 NEXT M
190 NEXT L
200 DATA 0.707, 0.707, 0, —0.707, 0.707, 0, 0, 0,1
210 DATA 1, 0, 0, 0, 1, 0, 30, —10, 1
220 DATA 0.5, 0, 0, 0, 0.3, 0, 0, 0, 1
230 LET I=0
240 GO SUB 1200
```

Naredbama 70 — 120 određuje se početni položaj temena trougla. Temena imaju sledeće koordinate: A(50,0), B(150,0), C(100,30).

Naredbama 130 — 220 određuje se sadržaj transformacionih matrica.

Rotaciju određuje ugao rotacije = —45°.

Matrica rotacije je:

$$\begin{bmatrix} 0.707 & 0.707 & 0 \\ -0.707 & 0.707 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Translaciju određuje Tx=30 i Ty=—10.

Matrica translacije je:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 30 & -10 & 1 \end{bmatrix}$$

Promenu veličine lika određuje $S_x=0.5$ i $S_y=0.3$

Matrica promene veličine je:

$$\begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Programskom rutinom SUB 1200 vrši se prikaz početnog položaja trougla na ekranu:

```
300 REM POJEDINACNE TRANSFORMACIJE
310 FOR I=1 TO 3
320 GO SUB 1000
330 GO SUB 1200
340 NEXT I
```

```
 990 REM MNOZENJE VEKTOR MATRICA
1000 FOR J=1 TO 3
1010 FOR K=1 TO 3
1020 LET A(I+1, J, K) = 0
1030 FOR P=1 TO 3
1040 LET A(I+1,J,K)=A(I+1,J,K)+
     A(I,J,P)*T(I+1,P,K)
1050 NEXT P
1060 NEXT K
1070 NEXT J
1080 RETURN
```

```
1190 REM PRIKAZ TROUGLA
1200 PLOT   A(I+1, 1, 1)— A(I+1, 1, 2)
1210 DRAW   A(I+1, 2, 1)— A(I+1, 1, 1),
            A(I+1, 2, 2)— A(I+1, 1, 2),
1220 DRAW   A(I+1, 3, 1)— A(I+1, 2, 1),
            A(I+1, 3, 2)— A(I+1, 2, 2),
1230 DRAW   A(I+1, 1, 1)— A(I+1, 3, 1),
            A(I+1, 1, 2)— A(I+1, 3, 2)
1240 RETURN
```

Naredbama 300 — 340 se u FOR . . . NEXT petlji sa upravljačkom promenljivom I izvršavaju programske rutine 1000 i 1200. Kada promenljiva I ima vrednost 1, obavi se prva transformacija i prikaz trougla posle prve transformacije, a kada je promenljiva I=2, druga transformacija i prikaz posle nje, itd.

Programska rutina 1000 obavlja osnovnu transformaciju koordinate A(I, . , .) u A(I+1, . , .). Osnovna transformacija se obavlja operacijom množenja vektora koordinata A(I,J,.) sa matricom transformacije T(I+1, . , .). Postupak množenja se obavlja naredbama 1010—1060. Kako se transformacija obavlja za sva tri temena trougla, navedeni skup naredbi je u okviru FOR . . . NEXT petlje čija upravljačka promenljiva J uzima vrednosti 1, 2, 3 (teme A, B, C).

Programska rutina 1200 vrši prikaz trougla na ekranu. Prikaz zavisi od promenljive I koja definiše koordinate za koje se prikaz vrši:

```
400 REM POVEZANA TRANSFORMACIJA
410 FOR L=2 TO 3
420 GO SUB 2000
430 LET P=L+1
440 IF P=4 THEN LET P=2
450 GO SUB 2200
460 NEXT L
470 LET I=1
480 GO SUB 1000
490 INK 2
500 GO SUB 1200
505 INK 0
510 STOP
```

```
1990 REM MNOZENJE MATRICA
2000 FOR M=1 TO 3
2010 FOR N=1 TO 3
2020 LET T(1, M, N) = 0
2030 FOR P = 1 TO 3
2040 LET T(1, M, N) = T(1, M, N)
     + T(L, M, P) * T(L+1, P, N)
2050 NEXT P
2060 NEXT N
2070 NEXT M
2080 RETURN
```

```
2190 REM PRENOS MATRICE
2200 FOR M=1 TO 3
2210 FOR N=1 TO 3
2220 LET T(P, M, N) = T(1, M, N)
2230 NEXT N
2240 NEXT M
2250 RETURN
```

Naredbama 410—460 se u FOR . . . NEXT petlji sa upravljačkom promenljivom L izvršavaju programske rutine 2000 i 2200. Kada promenljiva L ima vrednost 2, ona obavlja i množenje prve i druge matrice transformacije, kada je L=3 — množenje prethodnog rezultata i treće transformacione matrice. Takođe se uz pomoć programske rutine 2200 matrica T(1, . , .) prenosi na odgovarajuće mesto u matrici T da bi se množenje moglo nastaviti ili da bi se prešlo na drugu rutinu 1000 (izračunavanje novih koordinata).



Pokretanje grafičkih simbola

8.1. POKRETANJE PO EKRANU

U najvećem broju programa igara sa kojima ste se sigurno sreli, grafički simboli iz kojih se formira neka figura — kreću se. Kretanje može biti pod našom kontrolom. Kada pritisnemo na odgovarajuće dirke pomeramo figuru levo ili desno, gore ili dole. Neke figure u programima igara se kreću "same", isključivo rukovođene procesima koji se dešavaju u programu, a bez mogućnosti da mi na to utičemo. Kretanje figure u oba slučaja se programira na isti način.

Sledeći program pokreće figuru po ekranu koristeći funkciju TAB naredbe PRINT:

```
10 LET A = INT (RND * 25) + 1
20 PRINT TAB A; CHR$(141); CHR$(142)
30 POKE 23692,—1
40 GO TO 10
```

(naredba 30 POKE 23692,—1 omogućava da ce program ne zaustavlja kada se napuni ceo ekran televizijskog aparata u očekivanju naredbe SCROLL). Pozicija na kojoj se pojavljuju grafički simboli čiji su kôdovi 141 i 142 određuje se naredbom 10 i na slučajan način kao funkcija od RND.

Ovakvo pokretanje figure je svakako interesantno, ali može da služi samo za objašnjenje rada naredbe PRINT TAB. U programima igara najčešće imamo kombinaciju nepokretne slike, koja predstavlja predeo gde se igra dešava, i većeg broja pokretnih figura koje se kreću, ali stalno u okviru ekrana. Kretanja tog tipa moraju

se definisati naredbom PRINT AT zato što se njom definišu prikazi na željenim pozicijama (istog) ekrana.

Pokretanje figure se, ako se ograničimo samo na jedan pravac, uvek sastoji iz dve operacije, i to (I) prikaza na željenoj poziciji, i (II) brisanje slike figure na poziciji na kojoj se ona ranije nalazila. Pozicije na kojima figura treba da izađe najčešće se izražavaju preko sadržaja odgovarajućih promenljivih.

U sledećem primeru pomeramo grafički simbol čiji je kôd 143 duž linije numerisane sa 5. Posle prikaza grafičkog simbola na poziciji 5,X na istoj poziciji se prikazuje blanko znak, čime se vrši brisanje pred sledeći prikaz:

```
10 LET X=0
20 PRINT AT 5,X;CHR$(143)
30 PRINT AT 5,X;" "
40 LET X=X+1
50 GO TO 20
```

Pri izvođenju programa došlo je do greške. Pomeranjem grafičkog znaka upravlja promenljiva X koja uzima vrednosti 0, 1, 2, . . . ,30, 31, 32 . . . Kao što znamo, na ekranu u pravcu X—ose postoje 32 pozicije numerisane sa 0—31. Ako se premaši ta vrednost, dobiće se poruka o greški. Možemo zato programu dodati naredbu:

```
45 IF X > 31 THEN GO TO 10
```

čime svaki put, kada X dobije sadržaj koji "izlazi" sa ekrana, ponovo počinjemo sa kretanjem grafičkog simbola počev od leve margine ekrana.

Drugi problem koji smo uočili je sam prikaz simbola. U programu se odmah posle prikaza (naredba 20) vrši brisanje (naredba 30), tako da se ima utisak da figura treperi.

Loš utisak treperenja možemo otkloniti obezbeđenjem dužeg vremena prisustva prikaza figure na ekranu. To se može obezbediti ubacivanjem FOR . . . NEXT petlje koja ne vrši nikakvu funkciju ili upotrebom naredbe PAUSE:

```
10 LET X=0
20 PRINT AT 5,X; CHR$(143)
22 FOR I=1 TO 24
24 NEXT I
30 PRINT AT 5,X; " "
40 LET X=X+1
45 IF X > 31 THEN GO TO 10
50 GO TO 20
```


Ova verzija programa radi svakako bolje nego prethodna (treperenja više nema), ali je ceo proces dosta usporen. Taj problem rešava sledeći program:

```
10 LET X=0
20 LET P=X
30 LET X=X+1
40 IF X > 31 THEN LET X=0
50 PRINT AT 5,P;" "; AT 5,X; CHR$(143)
60 GO TO 20
```

U ovaj program je, pored promenljive X, uvedena i promenljiva P. Dok je promenljiva X i dalje apscisa prikaza grafičkog simbola, promenljiva P preuzima ulogu apscise blanko znaka. Pogledajmo kako se iz koraka u korak menjaju X i P:

Korak	P	X
1	0	1
2	1	2
3	2	3
4	3	4
30	29	30
31	30	31
32	31	0
33	0	1

Sadržaj promenljive P je uvek za 1 manji od sadržaja promenljive X. Ova verzija programa je veoma slična onoj od koje smo pošli, ali razlika je ipak bitna. Ovde naredbom 50 prvo vršimo brisanje prikaza figure na poziciji X—1, a odmah potom prikazujemo figuru na poziciji X. Posle toga se obavljaju operacije 60, 20, 30 i 40 i za sve to vreme figura stoji na poziciji X. Na taj način izbegnut je efekat treperenja i nepotrebno povećanje trajanja izvođenja programa koje proizilazi iz parazitske FOR . . . NEXT petlje.

Vidimo da sadržaj promenljive P uvek kasni za sadržajem promenljive X. Kada to znamo, možemo izbaciti promenljivu P iz programa. U tom slučaju ćemo, umesto prikaza grafičkog simbola, vršiti prikaz na poziciji X prvo blanko znaka, a potom grafičkog simbola:

```
10 LET X=0
20 LET X=X+1
30 IF X > 30 THEN LET X=0
40 PRINT AT 5,X;" ";CHR$(143)
50 GO TO 20
```

U naredbi 30 kao granična vrednost za X uvedeno je 30 umesto 31 — zato što sada odjednom prikazujemo dva znaka. Kada pokrenete izvođenje ovog programa, primetićete, kada se prvi put popuni red poslednja 31.-a, pozicija ostaje stalno zatamnjena. To je posledica prikaza u grupi. Izbeći ćemo to ubacivanjem nove naredbe:

```
45 IF X=0 THEN PRINT AT 5,31;" "
```

Isti zadatak možemo urediti i korišćenjem FOR . . . NEXT petlje:

```
10 FOR X=0 TO 30
20 PRINT AT 5,X;" ";CHR$(143)
30 NEXT X
40 PRINT AT 5,31;" "
50 GO TO 10
```

Kretanje opisano na ovakav način obavlja se brzo, daje lep utisak i predstavlja jednostavnu i pouzdanu programsku rutinu. Sada ćemo pokušati da ukažemo na elemente potrebne da bi se kretanje figure vršilo pod komandom.

Komande za kretanje se najčešće prihvataju preko INKEY$. Objašnjavajući kako radi INKEY$, rekli smo da, ako na primer pritisnemo dirku K, tada je INKEY$="K", a ako se ne pritisne ništa, INKEY$=" " (prazan string). Kad unos vršimo na ovaj način nije potrebno pritiskati dirku ENTER, pa je ceo postupak unosa brz.

Neke dirke se češće od ostalih koriste za pomeranje figura na ekranu. Tako, ako je pomeranje desno-levo, najčešće se koriste dirke Z i M. One se nalaze u poslednjem redu tastature: Z sa leve i M sa desne strane. Pritiskom na dirku Z vrši se pomeranje ulevo, a pritiskom na dirku M udesno.

Ako se pomeranje vrši i gore—dole, izbor često pada na dirke 5, 6, 7 i 8. One imaju ta značenja u drugom režimu rada: 5 se koristi za levo, 6 nadole, 7 nagore, a 8 nadesno.

Ako za koordinate figure uzmemo x,y i ako želimo da se vrši pomeranje figure pomoću dirki 5, 6, 7 i 8, u programu će se pojaviti i sledeće naredbe:

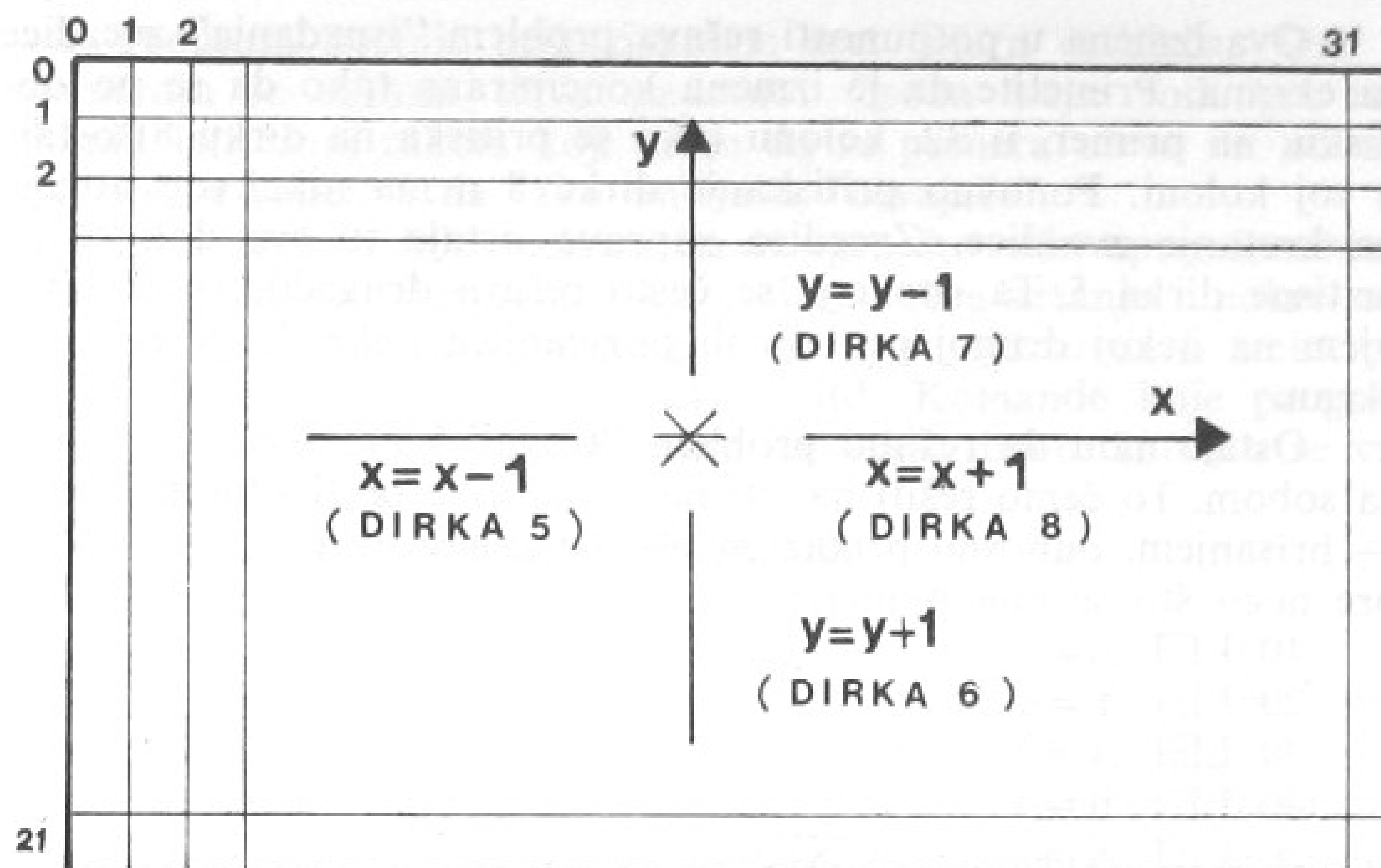
```
IF INKEY$ = "8" THEN LET X=X+1
```

Ako je pritisnuta dirka 8, sadržaj X se povećava za 1. Ako posle toga uradimo:

```
PRINT AT Y,X; CHR$(143)
```

prikaz će biti dat na sledećoj poziciji.

Sledeći program vršiće prikaz i pomeranje zvezdice "*" po ekranu zavisno od toga da li pritiskate 5, 6, 7 i 8.:



Sl. 8.1 — Pomeranje znaka "*" po ekranu uz korišćenje dirki 5, 6, 7 i 8

```
10 LET X=15
20 LET Y=10
30 PRINT AT Y, X;" * "
40 IF INKEY$ = "5" THEN LET X=X—1
50 IF INKEY$ = "8" THEN LET X=X+1
60 IF INKEY$ = "7" THEN LET Y=Y—1
70 IF INKEY$ = "6" THEN LET Y=Y+1
80 GO TO 30
```

Komandujući kretanjem zvezdice po ekranu vidimo da ona ostavlja trag za sobom. Takođe, ako dođemo do kraja ekrana, dobićemo poruku o greški. To je posledica postojanja 22 vrste na ekranu numerisane sa 0 — 21 i 32, kolone numerisane 0 — 31. Kao i ranije, ograničenje toga tipa moraćemo da unesemo u program:

```
40 IF INKEY$ = "5" AND X>0 THEN
   LET X=X—1
50 IF INKEY$ = "8" AND X<31 THEN
   LET X=X+1
60 IF INKEY$ = "7" AND Y>0 THEN
   LET Y=Y—1
70 IF INKEY$ = "6" AND Y< 21 THEN
   LET Y=Y+1
```

Ova izmena u potpunosti rešava problem "ispadanja" zvezdice sa ekrana. Primetite da je izmena koncipirana tako da se po dolasku, na primer, u 32. kolonu (ako se pritiska na dirku 8) ostaje u toj koloni. Ponovno pritiskanje dirke 8 nema nikakvog uticaja na kretanje zvezdice. Zvezdica zapravo ostaje tu sve dok se ne pritisne dirka 5. Ta situacija se često rešava drugačije, pojavljivanjem na nekoj drugoj poziciji ili pozivanjem neke druge slike na ekran.

Ostaje nam da rešimo problem "traga" koji zvezdica ostavlja za sobom. To ćemo rešiti na isti način kao i u prethodnom slučaju — brisanjem, odnosno prikazom blanko znaka na poziciji zvezdice pre nego što se ona pomeri:

```
10 LET X=0
20 LET Y=0
30 LET A=X
40 LET B=Y
50 LET X = X – (INKEY$ = "5" AND X>0)
             + (INKEY$ = "8" AND X<31)
60 LET Y = Y – (INKEY$ = "7" AND Y>0)
             + (INKEY$ = "6" AND Y<21)
70 IF A<>X OR B<>Y THEN
   PRINT AT B,A; " "
80 PRINT AT Y,X; "*"
90 GO TO 30
```

Izrazi u zagradama, kao što je:

(INKEY$ = "5" AND X>0)

jesu logički izrazi čija je vrednost 1, ako je uslov u zagradi zadovoljen, odnosno 0, ako uslov u zagradi nije zadovoljen. U navedenom primeru to znači "ako je pritisnuta dirka 5 i ako je X>0" uslov je zadovoljen; zagrada ima vrednost 1 i ta jedinica se oduzima od X.

8.2. IGRA "NAPADAČI"

U sleđećoj igri videćemo praktično kako se koriste elementi vezani za pomeranje figura koje smo obradili u prethodnom poglavlju.

Ovo je jedna od mnogih verzija igara tipa SPACE INVADERS. Jasno je da je igra što više uprošćena, da je razumne veličine i lako razumljiva. Znamo da se većina programa igara piše u mašinskom jeziku, a ne u BASIC-u, zbog poznatih prednosti prvog (brzina izvođenja, zauzeće memorije itd.).

Program počinje prikazivanjem šest figura osvajača. One se pomeraju po ekranu sleva nadesno. Osoba koja vodi igru raspolaže lanserom raketa koji može da se pomera ulevo ili udesno da lansira rakete i da tako uništava "osvajače".

Iako nam je bila namera da se igra što više uprosti, računar obavlja čitav niz postupaka. To podrazumeva crtanje i pokretanje napadača, uvid u stanje tastature, da bi se videlo da li treba pokretati lansere ili ispaljivati rakete, itd. Komande koje program prihvata vrši se pritiskanjem samo na jednu dirku, kao što se vidi na sledećoj tabeli:

dirka M — pomera lanser udesno,
dirka Z — pomera lanser ulevo,
dirka N — ispaljuje raketu.

Pritiskanjem drugih dirki za vreme izvođenja programa nema efekta. Lako je izmeniti program da prihvata komande i preko nekih drugih dirki, ako je to za vas iz nekih razloga prihvatljivije.

Kada se raketa ispali, program treba da je prikaže na njenom putu. Prilikom ispaljivanja treba stvoriti efekat da se raketa i "napadači" kreću u isto vreme. Te dve akcije se ne mogu izvesti istovremeno, pošto računar ima samo jedan procesor koji izvršava naredbe u sekvenci. Ta iluzija se stvara simultanim kretanjem prvo "napadača", potom rakete, sve malo po malo, tako da se smatra da se kreću istovremeno.

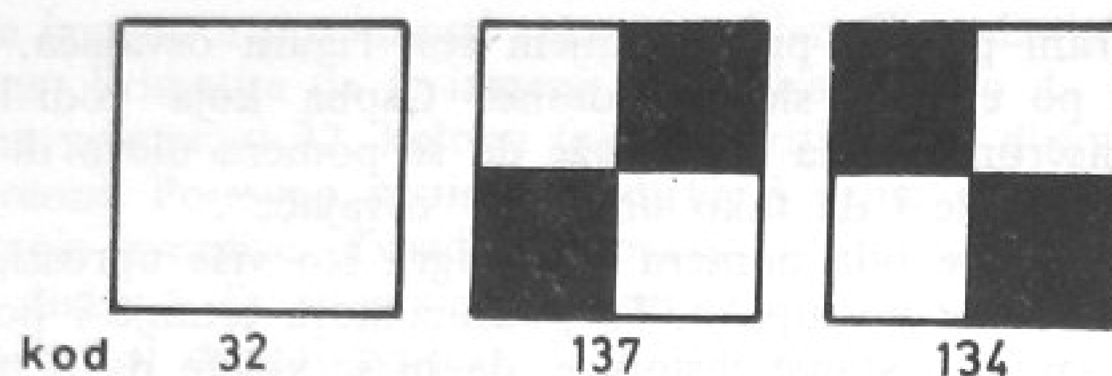
U fazi inicijalizacije obavljaju se sledeće operacije:

1. postavljanje početnih pozicija za "napadače",
2. postavljanje početnih pozicija za lansere raketa,
3. brisanje ekrana,
4. prikaz "napadača"
5. prikaz lansera.

Pri postavljanju početnih pozicija koristićemo dve promenljive sa strukturom niza, od kojih svaka ima po šest elemenata, da bi se pratile pozicije svakog od šest "napadača" na isti način. Niz koji se zove "R" sadrži broj vrste, a "C" broj kolone. Korišćenjem tih nizova, početne pozicije "napadača" biće uspostavljene sa:

```
40 FOR K=1 TO 6
50 LET R(K)=2
60 LET C(K)=4*K
70 NEXT K
```

Grafički karakteri koji se koriste prikazani su na slici: svaki "napadač" se pravi od tri znaka čiji su kodovi 32,137 i 134.' "Napadač" se prikazuje na sledeći način:

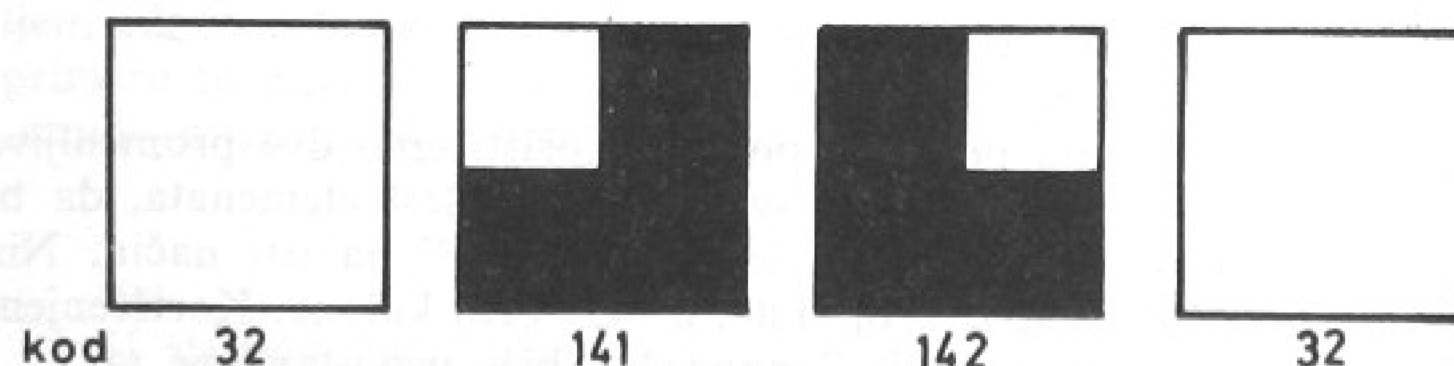


Sl. 8.2 — Figura "napadača"

Blanko znak na levoj strani se uvodi zato da se automatski blankira pozicija koju "napadač" napušta pri kretanju sleva nadesno:

```
1000 FOR K=1 TO 6
1020 LET C(K) = C(K)+1
1030 IF C(K) < 31 THEN GO TO 1060
1032 PRINT AT R(K), C(K)—1; "  "
1040 LET R(K) = R(K) + 1
1050 LET C(K)=1
1060 PRINT AT R(K), C(K)—1; "  "
1062 PRINT AT R(K), C(K); CHR$ (137)
1064 PRINT AT R(K), C(K)+1; CHR$ (134)
1110 NEXT K
```

Raketni lanser se pozicionira na najnižu vrstu ekrana. Znakovi sa kodovima 32,141,142 i 32 koriste se da bi se predstavio lanser, pošto on može da se pomera sleva nadesno. Rakete se predstavljaju znakom čiji je kôd 94.



Sl. 8.3 — Figura lansera raketa

U glavnom delu programa prihvataju se komande sa tastature korišćenjem INKEY$ naredbe. Zavisno od dirke koja se pritisne zove se neka od rutina (ili se ne radi ništa ako se pritisne pogrešna dirka).

Sledeća tabela daje listu rutina koje ovaj program koristi.


```
 10 REM NAPADAC
 20 REM *** INICIJALIZACIJA ***
 30 DIM R(6):DIM C(6)
 40 FOR K=1 TO 6
 50 LET R(K)=2
 60 LET C(K)=4*K
 70 NEXT K
 80 LET S=0
 90 LET LC=20
 95 LET R$=" "+CHR$ (141)+CHR$
(142)+" "
 100 CLS
 110 GO SUB 1000
 120 GO GUB 2000
 220 REM *** GLAVNA PROGRAMSKA P
ETLJA ***
 230 LET C$=INKEY$
 240 IF C$="M" THEN GO SUB 5000
 250 IF C$="Z" THEN GO SUB 6000
 260 IF C$="N" THEN GO SUB 7000
 270 GO TO 110
 998 REM PROGRAMSKA RUTINA ZA PO
MERANJE
 999 REM I PRIKAZ NAPADACA
1000 FOR K=1 TO 6
1020 LET C(K)=C(K)+1
1030 IF C(K)<31 THEN GO TO 1060
1032 PRINT AT R(K),C(K)-1;"  "
1040 LET R(K)=R(K)+1
1050 LET C(K)=1
1060 PRINT AT R(K),C(K)-1;" "
1062 PRINT AT R(K),C(K);CHR$ (13
7)
1064 PRINT AT R(K),C(K)+1;CHR$ (
134)
```

```
1110 NEXT K
1120 RETURN
1999 REM PROGRAMSKA RUTINA ZA PR
IKAZ LANSERA RAKETE
2000 PRINT AT 20,LC-1;R$
2060 RETURN
4999 REM PROGRAMSKA RUTINA ZA PO
MERANJE DESNO
5000 IF LC=29 THEN GO TO 5030
5010 LET LC=LC+1
5020 GO SUB 2000
5030 RETURN
5999 REM PROGRAMSKA RUTINA ZA PO
MERANJE LEVO
6000 IF LC=1 THEN GO TO 6030
6010 LET LC=LC-1
6020 GO SUB 2000
6030 RETURN
6999 REM PROGRAMSKA RUTINA ZA IS
PALJIVANJE RAKETE
7000 LET NR=18
7020 PRINT AT NR,LC;CHR$(94)
7040 LET NR=NR-1
7050 IF NR=R(6)-1 THEN GO TO 707
0
7060 GO TO 7020
7070 PRINT AT NR+1,LC;CHR$ (94)
7080 RETURN
```

Sl. 8.4 — Program "Napadači"

Pokušajte da dodate programu sistem za praćenje rezultata, a takođe i test za uništavanje napadača:

Početak rutine	Rutina
1000	prikaz i pomeranje napadača
2000	prikaz lansera
5000	pomeranje lansera desno
6000	pomeranje lansera levo
7000	ispaljivanje raketa i pomeranje "napadača"



Korisnikova grafika

9.1. DEFINISANJE GRAFIČKIH SIMBOLA

Pored skupa grafičkih simbola kojima SPECTRUM raspolaže i koje možete koristiti kada se nalazite u "grafik" režimu rada, postoji i mogućnost da sami definišete prema želji vaše, tj. korisničke simbole. Simboli definisani na ovaj način zovu se skraćeno UDG simbolima, odnosno UDG grafikom. (UDG je skraćenica od engleskog User Defined Graphics.)

Znamo da se ekran sastoji od 22 × 32 pozicije koje smo već koristili, na primer, kada smo želeli da izvršimo prikaz PRINT naredbom na određenoj poziciji ekrana. Tako:

PRINT AT 10,20;"*"

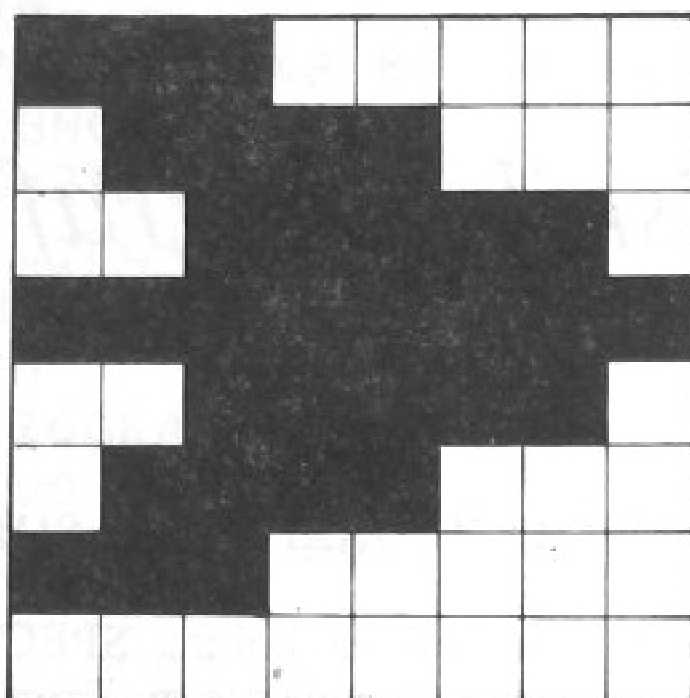
prikazuje zvezdicu u proseku 11. vrste i 21. kolone ekrana. Svaka od ovih pozicija sastoji se od 8 × 8 ćelija koje se (na engleskom jeziku) zovu Pixel. Uprošćeno govoreći, svaku od ovih ćelija možemo smatrati kao mrežu koja se sastoji od 8 × 8 kvadrata kojima programski možemo definisati stanje — svetao ili taman.

Odlučimo se sada da UDG simbol koji želimo da definišemo predstavlja raketu.

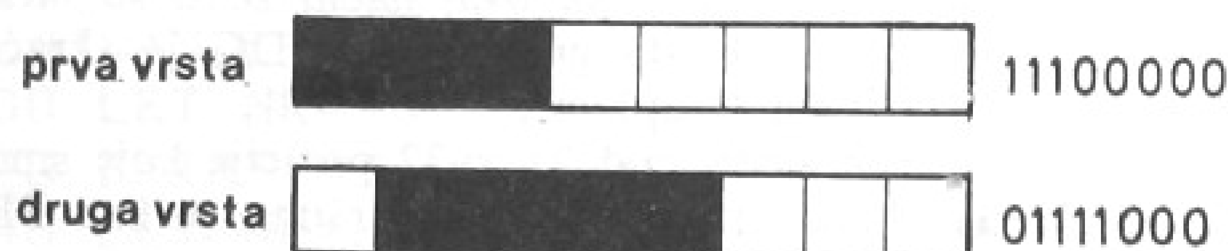
Prvo što ćemo uraditi to je da nacrtamo izgled simbola (sl. 9.1). Potom ćemo prema postojećoj šemi odrediti binarni broj koji odgovara svakoj posebnoj vrsti. Šta su binarni brojevi — sada nećemo objašnjavati, ali ako se prvi put srećete sa tim pojmom — recimo da pored decimalnog brojnog sistema, sa kojim se srećemo u svakodnevnom životu i kod koga se brojevi formiraju ciframa od 0—9, postoje i drugi brojni sistemi. Jedan od tih je i binarni sistem

koji se veoma koristi kod računara. Kod njega se brojne vrednosti izražavaju ciframa 0 i 1. U ovom slučaju nije nam neophodno da dobro poznajemo binarne brojeve, pošto se sa slike može lako odrediti kojoj vrsti koji binarni broj pripada.

Pogledajmo prvu i drugu vrstu (sl. 9.2).



Sl. 9.1 — Figura rakete



Sl. 9.2 — Prve dve vrste figure rakete i odgovarajući binarni brojevi

Prvoj vrsti dodeljen je broj 11100000. Svakoj zatamnjenoj ćeliji odgovara cifra 1, a nezatamnjenoj 0. Pošto svakoj vrsti odgovara 8 ćelija, za svaku vrstu napisaćemo uvek 8 binarnih cifara.

Druga vrsta počinje jednom svetlom ćelijom, zatim dolaze četiri zatamnjene i tri svetle. U ovom slučaju binarni broj će biti 01111000. Na taj način dolazimo do 8 binarnih brojeva:

11100000
01111000
00111110
11111111
00111110
01111000
11100000
00000000

Da bismo mogli ovim sadržajem da se koristimo, neophodno ga je smestiti na neku lokaciju u memoriju. Možemo izabrati neku od dirki A—U i njoj dodeliti ovako definisan grafički simbol. Posle toga, pozivanjem na izabranu dirku, pozivaćemo i UDG simbol. Ova grafička operacija važi i posle pritiska na dirku NEW.

Odlučimo se da simbol rakete dodelimo slovu A. Postupak dodeljivanja vrši se naredbama POKE USR. Za svaku vrstu iz slike, odnosno za svaki binarni broj potrebno je jednom aktivirati naredbu POKE USR.

Kada kažemo POKE USR ”A”, vršimo određivanje lokacije u memoriji SPECTRUMA na koju će se smestiti sadržaj koji sledi naredbu. Ako kažemo POKE USR ”A”+1, mislimo na lokaciju koja neposredno sledi prethodnu.

Potrebno je još saopštiti računaru da su vrednosti koje se unose date u binarnom obliku. To se čini naredbom BIN.

U programu koji sledi pritiskom na dirke 5, 6, 7 i 8 pomeraćemo lik rakete, pri čemu će njen trag da se označava zvezdicama:

```
 10 LET X = 0
 20 LET Y = 0
 30 POKE USR ”A”, BIN 11100000
 40 POKE USR ”A” + 1, BIN 01111000
 50 POKE USR ”A” + 2, BIN 00111110
 60 POKE USR ”A” + 3, BIN 11111111
 70 POKE USR ”A” + 4, BIN 00111110
 80 POKE USR ”A” + 5, BIN 01111000
 90 POKE USR ”A” + 6, BIN 11100000
100 POKE USR ”A” + 7, BIN 00000000
110 PRINT AT Y,X;”A”
120 LET A=X
130 LET B=Y
140 LET X=X — (INKEY$ = ”5” AND X > 0)
            + (INKEY$ = ”8” AND X < 31)
150 LET Y=Y — (INKEY$ = ”7” AND Y > 0)
            + (INKEY$ = ”6” AND Y < 21)
160 IF A < > X OR B < > Y THEN
    PRINT AT B,A; ”*”
170 GO TO 110
```

Pri unosu programa slovo ”A” u liniju 110 unesite pozicionirani u grafički režim rada tastature. Tada ”A” iz linije 110 postaje grafički simbol rakete. Ova zamena znakova važi samo u ”grafik” režimu rada.

Naredbe 30 — 100 najčešće se formulišu kombinacijom FOR... NEXT petlje i naredbe DATA:

```
30 FOR I=0 TO 7
40 READ J
50 POKE USR "A" + I, J
60 NEXT I
70 DATA BIN 11100000, BIN 01111000,
        BIN 00111110, BIN 11111111,
        BIN 00111110, BIN 01111000,
        BIN 11100000, BIN 0
```

BIN 0 i BIN 00000000 imaju isto značenje.

Najčešće se UDG simboli sastoje od nekoliko ćelija. Na taj način se mogu definisati figure koje imaju jasno uočljivu formu. Figura čoveka koju ćemo definisati sastoji se iz dve ćelije. Za njeno definisanje biće nam potrebno 16 binarnih brojeva. To su:

00011000
00111100
01100110
01111110
00111101
00011001
00111101
01011011
01011001
11011000
00111100
00100100
00100100
01100110
00000000
00000000

Formiraćemo figuru (kao i ranije) korišćenjem FOR... HEXT petlje. Pošto je u pitanju figura koja je u dve ćelije, upravljačka promenljiva petlje uzimaće vrednosti 0—15:

```
10 FOR I=0 TO 15
20 READ J
30 POKE USR "A" + I,J
40 NEXT I
```



```
50 DATA BIN 00011000, BIN 00111100, BIN 01100110,
        BIN 01111110, BIN 00111101, BIN 00011001,
        BIN 00111101, BIN 01011011, BIN 01011001,
        BIN 11011000, BIN 00111100, BIN 00100100,
        BIN 00100100, BIN 01100110, BIN 0, BIN 0
70 CLS
80 PRINT AT 10, 20; "A"
90 PRINT AT 10, 22; "B"
100 PRINT AT 15, 10; "A"
110 PRINT AT 16, 10; "B"
```

Naredbe programa 10—15 služe za formiranje figure. Kada naredbom 80 PRINT budemo tražili da se izvrši prikaz "A", dobićemo na ekranu prikaz samo gornjeg dela figure čoveka. Sledeći deo figure čoveka, tj. njegov donji deo, vezan je za simbol "B". Da smo imali neku figuru iz četiri dela, treći deo bi se dobio pozivanjem na "C", četvrti pozivanjem na "D" itd.

Naredbama 100 i 110 prikazujemo celu figuru. Pošto je figura uspravna prikazuje se u dva reda:

PRINT AT Y, X; "(grafika A)"

PRINT AT Y + 1, X; "(grafika B)"

Ako je figura u horizontalnom položaju, prikazuje se sa:

PRINT AT Y, X; "(grafika A)"; "(grafika B)"

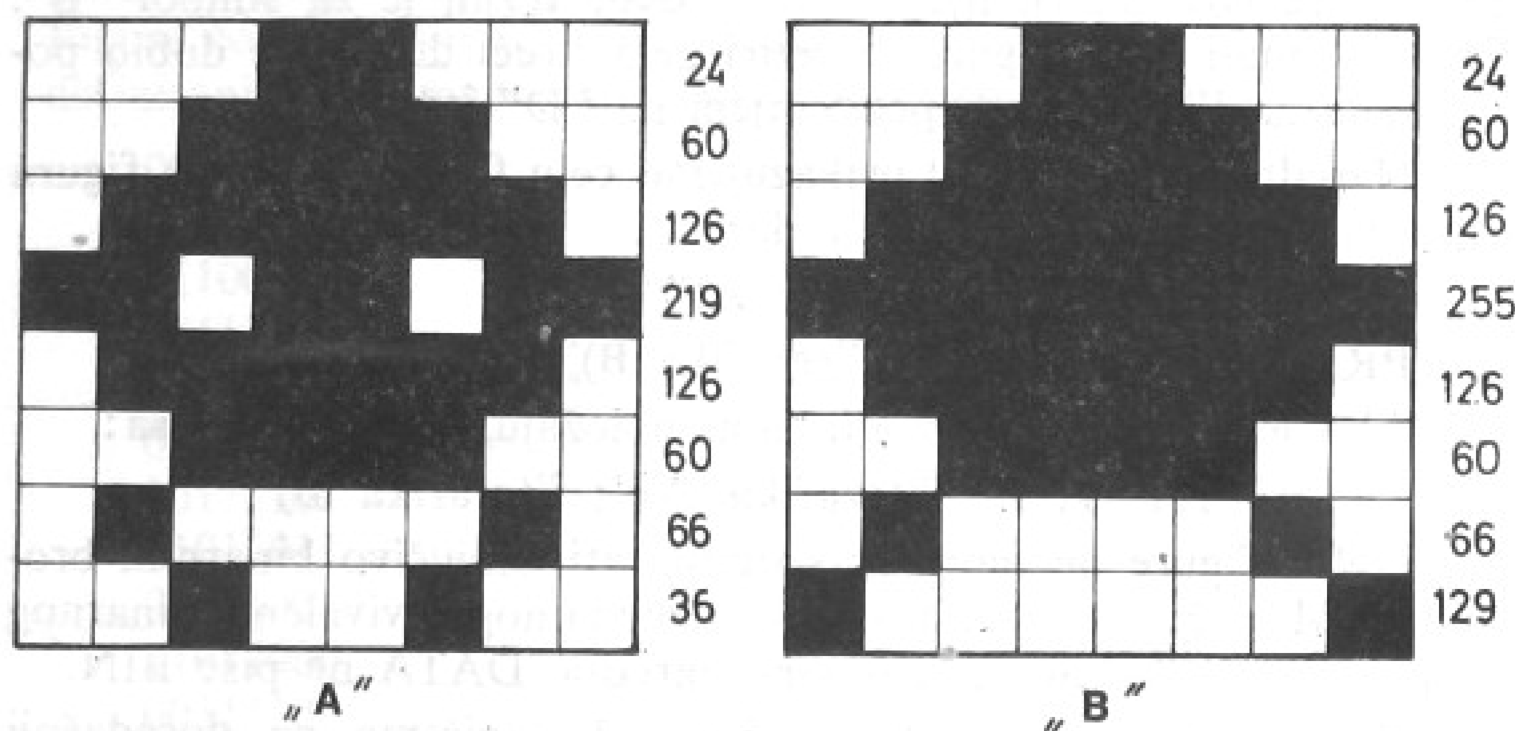
Izgled figure ne mora se specificirati isključivo binarnim brojevima. Moguće je to učiniti i preko decimalnog ekvivalenta binarnog broja. U tom slučaju se u okviru naredbe DATA ne piše BIN.

Figure sa sledeće slike možemo da opišemo na dosadašnji način, odnosno binarnim brojevima. Pored njih dati su i njihovi decimalni ekvivalenti (sl. 9.3).

00011000 = 24	00011000 = 24
00111100 = 60	00111100 = 60
01111110 = 126	01111110 = 126
11011011 = 219	11111111 = 255
01111110 = 126	01111110 = 126
00111100 = 60	00111100 = 60
01000010 = 66	01000010 = 66
00100100 = 36	10000001 = 129

Vidi se, na primer, da binarnom broju 00011000 odgovara decimalni broj 24. Sada će DATA lista biti dosta prostija za pisanje:

```
 10 FOR I = 0 TO 15
 20 READ J
 30 POKE USR "A" + I,J
 40 NEXT I
 50 DATA 24, 60, 126, 219, 126, 60, 66, 36,
         24, 60, 126, 255, 126, 60, 66, 129
 60 CLS
 70 PRINT AT 10, 10; "A"
 80 PRINT AT 10, 15; "B"
 90 PRINT AT 15, 5; "A"
100 PAUSE 10
110 PRINT AT 15, 5; "B"
120 PAUSE 10
130 GO TO 90
```



Sl. 9.3 — Figura čovečuljka. Uz svaku vrstu se nalaze odgovarajući decimalni brojevi

Naredbe 70 i 80 prikazuju prvu i drugu figuru na različitim pozicijama. I na ekranu možemo da uočimo glavne razlike među figurama. To su položaj nogu i oči.

U programskoj petlji 90 — 120 te dve figure prikazujemo na istoj poziciji. Vidimo da prvo prikazujemo figuru "A", posle toga figuru "B", itd. PAUSE se uvodi da bi se pojačao efekat pokretanja figure. Ako bismo izbacili naredbe 100 — 120, slike bi se tako brzo smenjivale da naše oko ne bi moglo da ih registruje. Čovečuljak ostavlja utisak da pokreće noge i da treperi očima. Na ovakvim postupcima se zasnivaju i složeniji postupci animacije.

Sledeći program simulira eksploziju:


```
 10 FOR I=0 TO 23
 20 READ J
 30 POKE USR "A" + I,J
 40 NEXT I
 50 DATA 0, 0,  8, 28, 28,  0,  0, 0,
         0, 8, 30, 34, 30, 12,  0, 0,
         0, 62, 65, 73, 73, 98, 28, 0
 60 PRINT AT 10, 10; "A"
 70 PAUSE 10
 80 PRINT AT 10, 10; "B"
 90 PAUSE 10
100 PRINT AT 10, 10; "C"
110 PAUSE 10
120 GO TO 60
```

9.2. "ZMAJEV LET"

Zmaj prikazan na sledećoj slici dat je u dva položaja. Na prvoj slici je sa rastavljenim krilima, dok su mu na sledećoj slici krila sastavljena. Svaka slika se sastoji iz četiri ćelije. Cilj nam je da napravimo program u kome ćemo pokretati figuru zmaja. Pri tom želimo da pri pokretanju maše krilima. Utisak pokretanja stvorićemo tako što ćemo pri pokretanju figure zmaja po ekranu na istoj poziciji uvek prikazati prvo jednu, a nešto docnije drugu sliku.

Definisanje izgleda figura i utisak pokretanja uradićemo odmah. Obe figure zauzimaju ukupno 8 ćelija, pa će nam za njihovo definisanje biti potrebno 8 × 8 binarnih vrednosti. Zato će upravljačka promenljiva uzimati vrednosti 0 — 63.

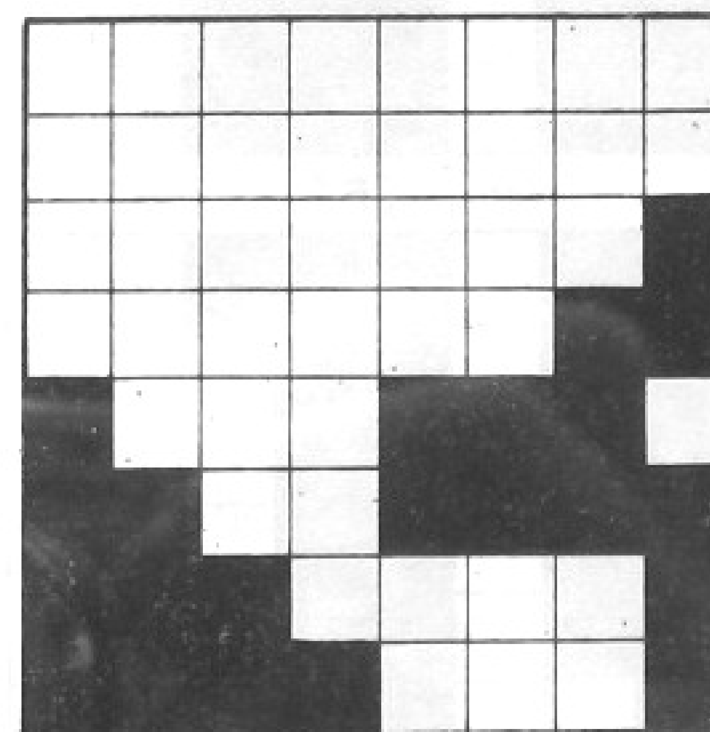
Pošto su u pitanju figure od četiri ćelije, koordinate Y i X kojima će se upravljati preko INKEY$ odnosiće se na levu gornju ćeliju. Gornja desna ćelija imaće koordinate Y, X+1, donja leva Y+1, X, a donja desna Y+1, X+1.

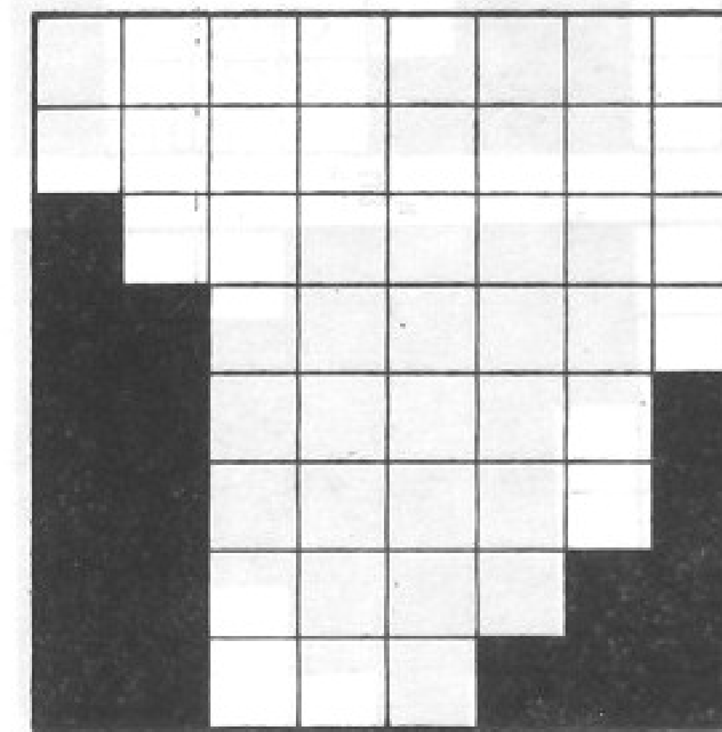
```
10 FOR I = 0 TO 63
20 READ J
30 POKE USR "A" + I, J
40 NEXT I
50 REM OTVORENA KRILA(Y,X)
60 DATA BIN 0, BIN 0 BIN 00000001,
        BIN 00000011, BIN 10001110,
        BIN 11001111, BIN 11100001,
        BIN 11110001
```

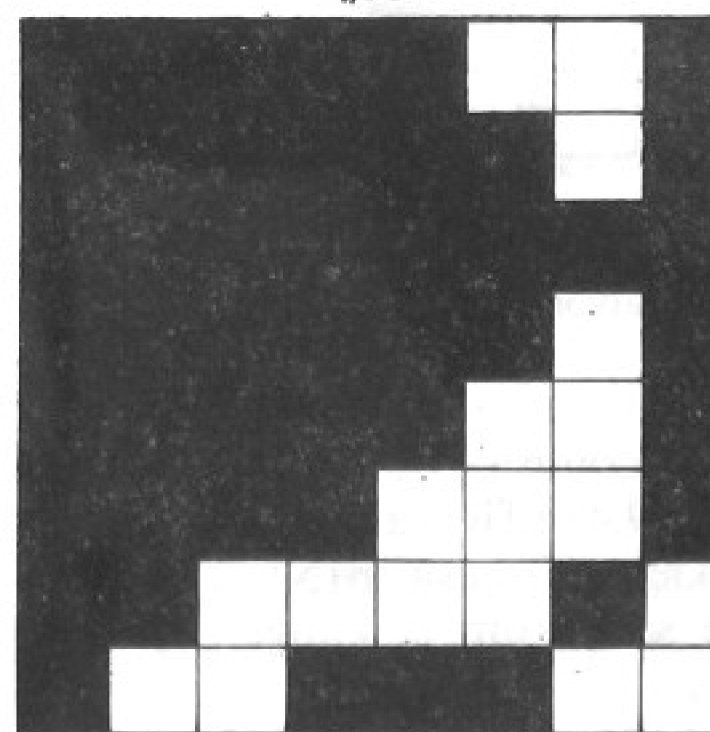
```
 70 REM OTVORENA KRILA (Y, X+1)
 80 DATA BIN 0, BIN 0, BIN 10000000,
        BIN 11000000, BIN 11000001,
        BIN 11000001, BIN 11000011,
        BIN 11000111
 90 REM OTVORENA KRILA (Y+1, X)
100 DATA BIN 11111001, BIN 11111101,
        BIN 11111111, BIN 11111101,
        BIN 11111001, BIN 11110001,
        BIN 11000010, BIN 10011100
110 REM OTVORENA KRILA (Y+1, X+1)
120 DATA BIN 11001111, BIN 11011111,
        BIN 11111111, BIN 11011111,
        BIN 11001111, BIN 1000111,
        BIN 00000011, BIN 00000001
130 REM ZATVORENA KRILA (Y,X)
140 DATA BIN 01100000, BIN 01100000,
        BIN 01100001, BIN 01100011,
        BIN 01101110, BIN 01101111,
        BIN 01100001, BIN 01110001
150 ZATVORENA KRILA (Y, X+1)
160 DATA BIN 00001000, BIN 00001000,
        BIN 10001000, BIN 11011000,
        BIN 11011100, BIN 11011100,
        BIN 11011100, BIN 11011100
170 ZATVORENA KRILA (Y+1, X)
180 DATA BIN 01111001, BIN 01111101,
        BIN 00111111, BIN 00111101,
        BIN 00111001, BIN 00010001,
        BIN 00000010, BIN 00010001,
        BIN 00000010, BIN 00011100
190 REM ZATVORENA KRILA (Y+1, X+1)
200 DATA BIN 11011100, BIN 11011000,
        BIN 11111000, BIN 11011000,
        BIN 11011000, BIN 10011000,
        BIN 00010000, BIN 00010000
300 CLS
310 REM PRIKAZ ZMAJA
320 PRINT AT 10,10;"A"; "B"
330 PRINT AT 11,10;"C"; "D"
340 PRINT AT 10,15;"E"; "F"
```

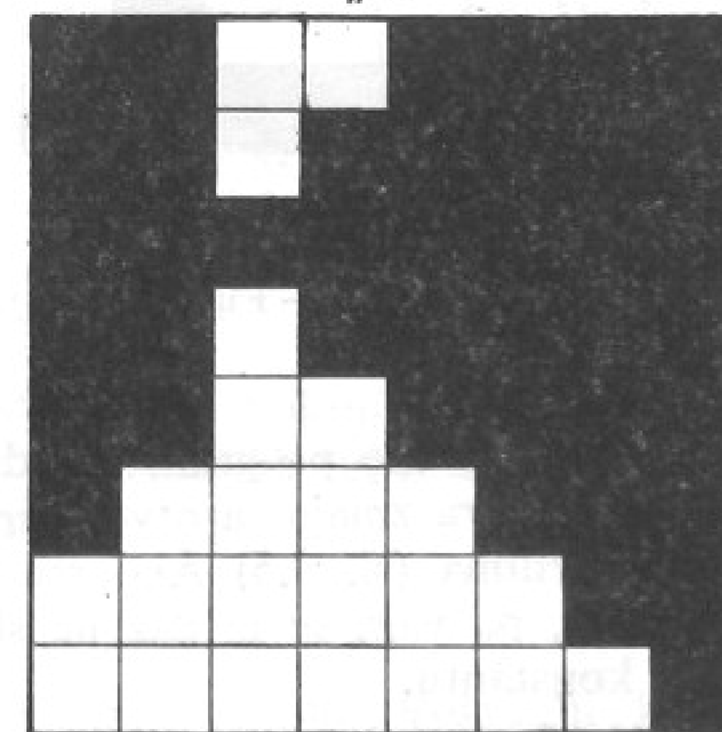


```
350 PRINT AT 11,15;"G"; "H"
360 PRINT AT 15,20;"A"; "B"
370 PRINT AT 16,20;"C"; "D"
380 PAUSE 10
390 PRINT AT 15,20;"E"; "F"
400 PRINT AT 16,20;"G"; "H"
410 PAUSE 10
420 GO TO 360
```

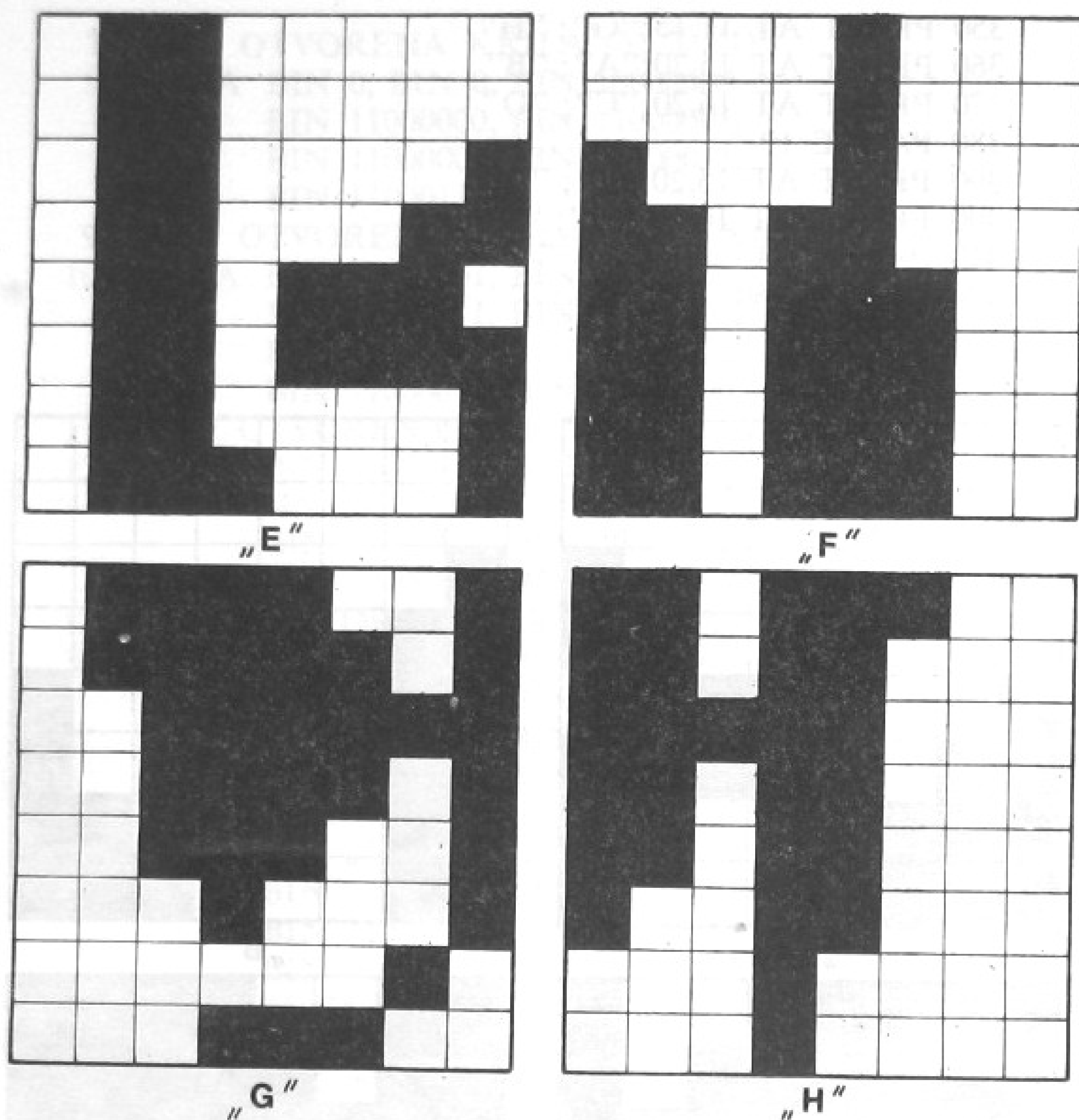

„A"


„B"


„C"


„D"

Sl. 9.4 — Figura "zmaja" sa otvorenim krilima



Sl. 9.5 — Figura "zmaja" sa zatvorenim krilima

Ako pustimo program u rad, na ekranu će se u desetom redu pojaviti figura zmaja sa otvorenim (sl. 9.4) i figura zmaja sa zatvorenim krilima (sl. 9.5). Ako ste prikazom unosa BIN konstanti pogrešili, pa vam se prikaz ne slaže sa slikom iz knjige, ispravite BIN konstantu.

U 15-om redu dat je prikaz zmaja koji maše krilima. Menjajte u naredbama 380 i 410 dužinu pauze i zaključite kako radi naredba PAUSE.

Sada možemo ići dalje i dopuniti program naredbama za pokretanje zmaja. Kao i ranije, koristićemo se dirkama 5, 6, 7 i 8 za pokretanje figure levo, dole, gore i desno. Te naredbe ćemo primiti preko INKEY$. Obrišite prvo naredbe 310 — 420:

```
310 LET X=15
320 LET Y=10
330 LET A=X
340 LET B=Y
350 LET X=X — (INKEY$ = "5" AND X > 0)
            + (INKEY$ = "8" AND X < 30)
360 LET Y=Y — (INKEY$ = "7" AND Y>0)
            + (INKEY$ = "6" AND Y< 20)
370 IF A < > X OR B < > Y THEN
    PRINT AT B,A;"   ";
    PRINT B+1, A;"   "
380 PRINT AT Y,X; "A"; "B"
390 PRINT AT Y+1,X;"C"; "D"
400 PAUSE 10
410 PRINT AT Y,X; "E"; "F"
420 PRINT AT Y+1,X; "G"; "H"
430 PAUSE 10
440 GO TO 330
```

9.3. FUNKCIJA SCREEN$

Korišćenjem funkcije SCREEN$ dobijamo znak koji se nalazi na specificiranoj poziciji ekrana. Uz funkciju SCREEN$ kao argumenti idu vrsta i kolona na ekranu u čijem se preseku nalazi znak koji nas interesuje. Naredbom:

```
PRINT AT 10,10; "A"
```

prikazaćemo na ekranu u 11-oj vrsti i 11-oj koloni slovo A. Ako želimo da nam se prikaže znak koji se nalazi na toj poziciji, korišćenjem naredbe SCREEN$ uradićemo:

```
PRINT SCREEN$ (10,10)
```

i prikazaće se ponovo znak A.

Funkciju SCREEN$ koristićemo u sledećem programu koji predstavlja uprošćenu verziju igre "Vožnja automobila" (sl. 9.6). Automobil se nalazi na fiksiranoj vertikalnoj poziciji, stalno u 13-oj vrsti ekrana. Korišćenjem dirki Z i M vrši se njegovo horizontalno

```
 10 REM VOZNJA AUTOMOBILA
 20 LET C=0
 30 LET T=0
 40 GO SUB 250
 50 LET A=10
 60 LET X=13
 70 LET Y=12
 80 LET K=INT (RND*2)
 90 LET A=A-(K=1 AND A>1)+(K=0
AND A<24)
 110 PRINT AT Y,X-1; INK 1;"C"
 120 PRINT AT 20,A; INK 2;"*";TA
B A+6;"*"
 130 PRINT
 140 POKE 23692,-1:PRINT
 150 PRINT INK 6; PAPER 2;AT 0,1
0;" REZULTAT JE ";T;"  "
 160 IF SCREEN$ (Y+1,X-1)="*" TH
EN GO TO 200
 170 LET X=X-(INKEY$="Z")+(INKEY
$="M")
 180 LET T=T+1
 190 GO TO 80
 200 PRINT AT Y,X-1; INK 1; "C"
 210 PRINT AT 6,6; FLASH 1; BRIG
HT 1;" SLETELI STE SA DRUMA!!"
 220 PRINT AT 8,10; FLASH 1; BRI
GHT 1; INK RND*7; PAPER 9;"  REZ
ULTAT JE ";T;" "
 230 BEEP .01,RND*20-RND*20
 240 GO TO 210
 250 FOR J=0 TO 7
 260 READ Z
 270 POKE USR "C"+J,Z
 280 NEXT J
 290 RETURN
```



```
 300 DATA BIN 00110110,BIN 00110
110,BIN 00111110,BIN 00010100,BI
N 00111110,BIN 00110110,BIN 0001
1100,0
```

Sl. 9.6 — Program "Vožnja automobila"

pomeranje ulevo i udesno. Ove komande se prihvataju naredbom INKEY$ (naredba 170). Utisak vožnje stvara stalno vertikalno pomeranje redova nagore. To je posledica naredbe 140 POKE 23692,—1. Slika puta je predstavljena zvezdama koje predstavljaju njegove ivice. Ivice puta se pomeraju ulevo i udesno na slučajan način.

Cilj igre je da se, komandujući kretanjem automobila čiji je izgled definisan naredbom 300 DATA, izbegne sletanje sa druma, odnosno udar u ivicu druma. Ta situacija se registruje preko funkcije SCREEN$. Ako je pozicija koja se nalazi neposredno ispred automobila "*", smatra se da je automobil sleteo sa druma. Tada se na ekranu pojavljuje poruka "SLETELI STE SA DRUMA".

Kod većine igara ova funkcija se koristi da bi se utvrdilo da li će doći do "sudara" dve figure.

Obratite pažnju, kada budete unosili program, da slovo C koje se pod navodnicama pojavljuje u linijama 110 i 200 unesete pozicionirani u grafički režim rada tastature.



Igre

10.1. IGRA "SLALOM"

Kao na svakoj skijaškoj stazi gde se vozi slalom, i ovde je cilj da skijaš prođe kroz što veći broj kapija. To se direktno odražava na njegov rezultat igre koji se stalno ispisuje u desnom gornjem uglu ekrana. Ako skijaš prođe kroz kapiju, odnosno između dve zastavice koje su u istoj vrsti, dobija 1 poen. Ako promaši kapiju, tada mu se rezultat umanjuje za 5 poena. Igra se prekida ako skijaš "udari u drvo" koje se nalazi na stazi. Figura skijaša može se pomerati samo ulevo i udesno. Kada se želi ići ulevo, potrebno je pritisnuti dirku Q. U tom slučaju prikazuje se figura skijaša sa skijama okrenutim ulevo. Ako se pritisne P, ide se udesno i to sa drugom figurom skijaša.

Program

U programu se koriste sledeći grafički simboli:

— Figura skijaša koji se kreće ulevo predstavlja se grafičkim simbolima koji su dodeljeni slovima "A", "B", "C" i "D".

Figura skijaša koji se kreće udesno predstavlja se grafičkim simbolima koji su dodeljeni slovima "E", "F", "G" i "H".

— Drvo, tj. figura jelke na skijaškoj stazi predstavlja se grafičkim simbolima koji su dodeljeni slovima "I", "J", "K" i "L".

— Leva i desna zastavica kapije imaju isti oblik i predstavljaju se grafičkim simbolima koji su dodeljeni slovima "M" i "N".


```
  10 REM SLALOM
  20 REM ----------------------------
  30 GO SUB 1000
  40 BORDER 7: PAPER 7: INK 0: C
LS
  50 LET X=16: LET T=0: LET C=80
  60 LET V=0
  70 REM ----------------------------
  90 REM IGRA
 100 LET A=INT (RND*15)+7: POKE 
23692,255
 110 IF POINT (X*8,159)=1 OR POI
NT ((X+1)*8,159)=1 OR POINT ((X+
2)*8,159)=1 THEN GO TO 400
 120 IF T/8<>INT (T/8) THEN GO T
O 160
 130 PRINT AT 21,A; INK 2;"M
  M"
 140 PRINT INK 7;CHR$ (A+40);TAB
 A; INK 2;"N        N"
 150 PRINT : GO TO 200
 160 IF T/3<>INT (T/3) THEN GO T
O 190
 170 PRINT AT 21,A; INK 4;"IJ":
PRINT TAB A; INK 4;"KL"
 180 PRINT : GO TO 200
 190 PRINT AT 21,0: PRINT : PRIN
T
 200 LET T=T+1: PRINT AT 0,28;V
 210 LET C$=INKEY$: IF C$<>"" TH
EN LET C=CODE C$
 220 IF C=81 THEN PRINT AT 0,X;"
AB": PRINT AT 1,X;"CD": IF X>6 T
HEN LET X=X-1
 230 IF C=80 THEN PRINT AT 0,X;"
EF": PRINT AT 1,X;"GH": IF X<28
```

```
THEN LET X=X+1
 240 PAUSE 5
 250 LET P=CODE SCREEN$ (2,0): I
F P=32 THEN GO TO 100
 260 IF X>P-41 AND X<P-33 THEN L
ET V=V+1: GO TO 100
 270 LET V=V-5: GO TO 100
 280 REM -------------------------
 290 REM IZGUBLJENA IGRA
 400 FOR I=-50 TO 50
 410 BEEP .01,I
 420 NEXT I
 430 IF INKEY$="" THEN GO TO 430
 440 GO TO 40
 980 REM -------------------------
 990 REM GRAFICKI SIMBOLI
1000 DATA 3,3,1,7,15,11,19,11,12
8,128,0,192,224,160,192,160
1010 DATA 6,3,2,2,3,2,21,10,148,
168,212,160,192,128,0,0
1020 DATA 1,1,0,3,7,5,3,5,192,19
2,128,224,240,208,200,208
1030 DATA 41,21,43,5,3,1,0,0,96,
192,64,64,192,64,168,80
1040 DATA 0,0,1,3,7,15,15,31,0,0
,0,128,192,224,224,240
1050 DATA 63,63,127,255,3,3,3,0,
248,248,252,254,128,128,128,0
1060 DATA 1,3,7,15,31,63,1,1,1,1
,1,1,1,1,1,1
1070 FOR I=0 TO 111
1080 READ A: POKE USR "A"+I,A
1090 NEXT I
1100 RETURN
```

Sl. 10.1 — Program igre ''Slalom''

— Grafički simboli se formiraju programskom petljom 1070 — 1090.

Obratite pažnju kada budete unosili program — da slova A—N koja se pod navodnicima pojavljuju u linijama 130, 140, 170, 220, 230 unesete pozicionirani u grafički režim rada tastature.

Inicijalizacije. Naredbama 30—60 vrši se inicijalizacija već opisane grafike i dodeljivanje početnih vrednosti promenljivim koje upravljaju igrom. To su:

— X je pozicija skijaša, i to horizontalna, pošto se figura uvek nalazi na vrhu ekrana (u fiksnim vrstama). Kao početna vrednost dodeljuje mu se 16, tj. pozicionira se na sredinu ekrana.

— T je broj pređenih linija u toku igre. Na početku je jednak nuli.

— V se koristi za praćenje rezultata igre. Na početku je jednako nuli.

— C je pravac pomeranja figure i može imati sadržaj 80 — desno, 81 levo. Na početku jednako je 80.

Igra. Pozicije drveća i kapija na stazi zavise od sadržaja slučajne promenljive A (naredba 100).

Kapije se postavljaju u slučaju da je sadržaj promenljive T deljiv sa 8 (n 120—150). U prvoj poziciji te linije upisaće se znak čiji je kôd A+40, čime se obeležava pozicija kapije na stazi. Ovo se koristi kasnije da bi se utvrdilo da li je skijaš prošao kroz kapiju ili ne.

Drveće se postavlja takođe zavisno od sadržaja promenljive T. Ako je T deljivo sa 3, postavlja se na poziciji A (n. 160, 170).

U okviru naredbe 100 javlja se i POKE 23692, 255 koji omogućava stalno pomeranje slike na ekranu, čime se izbegava odgovor na SCROOL.

Funkcijom POINT ispituje se da li je slobodna pozicija ekrana ispred figure skijaša (n. 110). Ako to nije slučaj, znači da je igrač ''udario'' u drvo ili u zastavicu kapije. Tada se ide na naredbe 290—420, gde se zvučnim signalom oglašava da je igra izgubljena. Igra se potom može započeti pritiskom na bilo koju dirku (n. 430—440).

Argumenti funkcije POINT koji se moraju pisati u zagradi jesu koordinate osnovnih ćelija pozicija ekrana (pixel). One mogu imati vrednosti X=0—255, Y=0—175. Efekat funkcije je sledeći: ako je boja označene pozicije ista kao boja središnjeg dela ekrana (PAPER) efekat je nula; ako je boja označene pozicije ista kao boja zapisa (INK) efekat je 1.

Naredbom 250 se ispituje da li se na liniji na kojoj je figura skijaša nalazi kapija. Ako je to slučaj, ispituje se da li je skijaš promašio kapiju. Tada se i ažurira rezultat igre.

10.2. IGRA "AUTO-PUT"

U ovoj igri treba da pomognete vozaču automobila koji je greškom uleteo na auto-put, i to na traku gde se sva ostala vozila kreću u pravcu suprotnom od njegovog. Cilj igre je preći što više "kilometara" bez udesa. Da biste izbegli vozila koja idu na vaš automobil stoje vam na raspolaganju dirke Q i P. Q se koristi za skretanje ulevo, a P udesno.

Program

Grafika. U programu se koriste sledeći grafički simboli:

— Figura automobila kojom upravljamo predstavlja se grafičkim simbolima koji su dodeljeni slovima "A", "B", "C" i "D".

— Figure ostalih automobila predstavljaju se grafičkim simbolima koji su dodeljeni slovima "E", "F", "G" i "H".

— Ivica puta se ne definiše korisničkom grafikom, već preko standardnih grafičkih simbola čiji su kodovi 133 i 138.

— Grafički simboli se formiraju programskom petljom 40 — 60.

Obratite pažnju kada budete unosili program — da slova A—H koja se pod navodnicima pojavljuju u linijama 210, 220, 350, 360, 420, 430, 440, 450 unesete pozicionirani u grafički režim rada tastature.

Inicijalizacije. Naredbama 30 — 160 vrši se inicijalizacija već opisane grafike i dodeljivanje početnih promenljivim. To su:

— X pozicija figura automobila kojom mi upravljamo. Ovo je samo horizontalna pozicija, pošto se figura automobila nalazi u fiksnoj vrsti na vrhu ekrana. Na početku je X=14, tj. pozicioniran je na sredinu ekrana.

— P se koristi za praćenje rezultata igre. Predstavlja broj pređenih kilometara bez sudara. Na početku je jednako nuli.

— N je nivo igre i može biti 1 ili 3. Njegova vrednost se dobija preuzimanjem (INPUT). Sadržaj ove promenljive određuje gustinu saobraćaja na auto-putu.

Igra. Pozicije automobila koji nam se kreće u susret jesu funkcija promenljive A čiji se sadržaj menja na slučajan način (naredba 200).


```
  10 REM AUTOPUT
  20 REM ------------------------
  25 REM GRAFICKI SIMBOLI
  30 DATA 2,55,55,63,55,55,7,5,5
,6,3,11,15,11,3,1,64,236,236,252
,236,236,224,160,160,96,192,208,
240,208,192,128
  36 DATA 1,3,11,15,11,3,3,3,3,3
,3,55,55,63,55,51,128,192,208,24
0,208,192,192,192,192,192,192,23
6,236,252,236,204
  40 FOR I=0 TO 63
  50 READ A: POKE USR "A"+I,A
  60 NEXT I
  70 INPUT "NIVO 1 ILI 2 ";N
  80 IF N<1 OR N>2 THEN GO TO 70
  90 IF N=2 THEN LET N=3
 150 LET X=14
 160 LET P=0
 190 REM -------------------------
 192 REM IGRA
 198 POKE 23692,255
 200 LET A=INT (RND*16) +7
 210 INK 6: PRINT AT 21,6;CHR$ (
133);: INK 1: PRINT TAB A;"EG";:
 INK 6: PRINT TAB 24; CHR$ (138)
 220 INK 1: PRINT TAB A;"FH"
 230 FOR I=1 TO N: PRINT : NEXT
I
 240 BEEP .001,14
 300 LET C$=INKEY$
 310 IF C$="" THEN GO TO 340
 320 IF C$="P" AND X<22 THEN LET
 X=X+1
 330 IF C$="Q" AND X>7 THEN LET
 X=X-1
```

```
340 LET P=P+1
350 INK 2: PRINT AT 0,X;"AC"
360 PRINT TAB X;"BD";TAB 22;P
370 IF POINT (X*8,159)=0 AND PO
INT ((X+1)*8,159)=0 AND POINT ((
X+2)*8,159)=0 THEN GO TO 190
380 REM -----------------------
390 REM KRAJ IGRE
400 FOR I=0 TO 5
410 BEEP .1,I
420 PRINT AT I,X-I;" ";AT I+1,X
-I-1,"A"
430 PRINT AT I,X+I+1;" ";AT I+1
,X+I+2;"C"
440 PRINT AT I+1,X-I;" ";AT I+2
,X-I-1;"B"
450 PRINT AT I+1,X+I+1;" ";AT I
+2,X+I+2;"D"
460 NEXT I
470 CLS
480 PRINT P;" KILOMETARA"
490 PAUSE 0
500 GO TO 150
```

Sl. 10.2 — Program igre "Auto-put"

Prikaz tih automobila i ivica puta vrši se naredbama 210—240. Tu se takođe, zavisno od nivoa igre, reguliše gustina saobraćaja na auto-putu.

Pravac pomeranja igračevog automobila prihvata se i sprovodi naredbama 300 — 360. Tu se istovremeno povećava sadržaj promenljive P za 1.

Funkcija POINT se koristi za ispitivanje da li je došlo do "sudara" sa nekim drugim automobilom. Ako se to desi, sudar se obeležava animacijom i zvukom, posle čega se ispisuje broj pređenih kilometara.

10.3. IGRA "PIRANE"

Teško će se neko spasti ako padne u reku punu krvoželjnih pirana. U ovom slučaju ne treba se odmah predavati, već tako voditi igru da se istraživač, koji je junak ove igre, izvuče živ.

Pirane koji se koriste u ovoj igri su toliko krvožedne da se čak međusobno proždiru. Figura istraživača može se pomerati pritiskanjem na dirke 5 — 8, čime se on može pomerati u sva četiri pravca (5 — levo, 6 — dole, 7 — gore i 8 — desno). Pirane se sa druge strane pomeraju tako da se što više približe istraživaču.

Zato istraživača treba pokretati tako da pokretanje navodi pirane da se međusobno proždiru.

U slučaju da izgubite igru, omotač ekrana promeniće sve moguće boje. Ako ste pobedili i uništili pirane, ispisaće se ukupan broj poteza koje ste povukli.

Igra se može nastaviti pritiskom na bilo koju dirku tastature.

Program

Grafika. U programu se koriste sledeći grafički simboli:

— Figuri pirane odgovaraju grafički simboli dodeljeni slovima "A", "B", "C" i "D". Pirana zauzima samo jednu poziciju, a kroz navedena četiri simbola predstavljena je u različitim pravcima kretanja: levo, dole, gore i desno.

— Figura istraživača predstavlja se grafičkim simbolom koji je pridružen slovu "E".

— Za omotač ekrana korišćen je standardni grafički simbol čiji je kôd 143.

Obratite pažnju kada budete unosili program da slova A—E koja se pod navodnicima pojavljuju u linijama 140, 170, 690, 800 i 900 unesete pozicionirani u grafički režim rada tastature.

Inicijalizacije. Naredbama 30 — 160 vrši se inicijalizacija već opisanih grafičkih znakova i dodeljivanje početnih vrednosti promenljivim koje upravljaju igrom.

— niz X$ sadrži x koordinate pirana,

— niz Y$ sadrži y koordinate pirana,

— p, čiji se sadržaj računa na slučajan način, predstavlja ukupan broj pirana. Može imati vrednost 20—30.

— promenljive X i Y (skalari) su koordinate istraživača. Početne pozicije pirana i istraživača se biraju na slučajan način.

— niz V sadrži broj mesta pomeranja u pravcu X ose.

— niz W sadrži broj mesta pomeranja u pravcu Y ose.

```
 10 REM PIRANE
 20 REM ---------------------------
 30 DIM X(30): DIM Y(30): DIM V
(4): DIM W(4)
 40 GO SUB 2000
 50 LET G=0: LET D=0: LET K=1
 60 PAPER 5: BORDER 5: INK 7: C
LS : INK 0
 70 FOR I=1 TO 30
 80 PRINT AT 1,I; INK 3; CHR$ (
143); AT 20,I; CHR$ (143)
 90 IF I<21 THEN PRINT AT I,1;
INK 3;CHR$ (143);AT I,30;CHR$ (1
43)
 100 NEXT I
 110 LET P=INT (RND*11)+20
 120 FOR I=1 TO P
 130 LET X(I)=INT (RND*28)+2: LE
T Y(I)=INT (RND*18)+2
 140 PRINT AT Y(I),X(I);"A"
 150 NEXT I
 160 LET X=INT (RND*28)+2: LET Y
=INT (RND*18)+2
 170 PRINT AT Y,X; INK 2; "E"
 180 PAUSE 0
 190 REM ---------------------------
 290 REM POMERANJE PIRANA
 300 FOR I=1 TO P
 310 IF X(I)=0 THEN GO TO 600
 320 PRINT AT Y(I),X(I); INK 7;"
"
 330 FOR J=1 TO 4
 340 IF ATTR (Y(I)+W(J),X(I)+V(J
))=42 THEN GO TO 800
 350 IF ATTR (Y(I)+W(J),X(I)+V(J
))=40 THEN GO TO 380
```



```
 360 NEXT J
 370 GO TO 410
 380 LET X(I)=0: LET G=G+1: BEEP
.05,I
 390 IF G=P-1 THEN GO TO 900
 400 GO TO 600
 410 IF Y(I)>Y THEN LET K=3: GO
TO 450
 420 IF Y(I)<Y THEN LET K=2: GO
TO 450
 430 IF X-1>=X(I) THEN LET K=4:
GO TO 450
 440 LET K=1
 450 LET X(I)=X(I)+V(K): LET Y(I
)=Y(I)+W(K)
 580 REM ---------------------------
 590 REM POMERANJE IGRACA
 600 IF INKEY$="" THEN GO TO 700
 610 LET C$=INKEY$
 620 FOR J=1 TO 4
 630 IF C$=CHR$ (52+J) THEN GO T
O 650
 640 NEXT J: GO TO 700
 650 IF ATTR (Y+W(J),X+V(J))=43
THEN GO TO 700
 660 IF ATTR (Y+W(J),X+V(J))=40
THEN GO TO 800
 670 PRINT AT Y,X; INK 7;" "
 680 LET X=X+V(J): LET Y=Y+W(J)
 690 PRINT AT Y,X; INK 2;"E": LE
T D=D+1
 700 IF X(I)>0 THEN PRINT AT Y(I
),X(I);CHR$ (143+K)
 710 NEXT I
 720 GO TO 300
 780 REM ---------------------------
```

```
790 REM IZGUBLJENA IGRA
800 PRINT AT Y,X; FLASH 1;"E"
810 FOR I=0 TO 7: BORDER I: BEE
P .5,I: NEXT I
820 PAUSE 0
830 RUN
880 REM ------------------------
890 REM DOBIJENA IGRA
900 PRINT AT Y,X; FLASH 1;"E"
910 PRINT AT 21,8;D;" POKRETA "
920 FOR I=-10 TO 10: BEEP  .1,I
: NEXT I
930 PAUSE 0
940 RUN
1980 REM ------------------------
1990 REM GRAFICKI SIMBOLI
2000 FOR I=0 TO 39
2010 READ A: POKE USR "A"+I,A
2020 NEXT I
2030 LET V(1)=-1: LET V(4)=1
2040 LET W(2)=1: LET W(3)=-1
2050 RETURN
2100 DATA 0,57,93,255,125,57,0,0
,124,16,56,124,124,92,56,16
2110 DATA 8,28,58,62,62,28,8,62,
0,156,186,255,190,156,0,0
2120 DATA 157,157,153,255,126,60
,60,255
```

Sl. 10.3 — Program igre "Pirane"

Igra. Pomeranje istraživača definisano je stanjem dirki (5 — 8) koje se prihvataju naredbom INKEY$ (n 600—730).

Pomeranje pirana se vrši u pravcu koji je najviše približava istraživaču. Pri tome se može desiti da se naiđe na drugu piranu, u kom slučaju je proždire (n 350, 380—400). Prisustvo pirana odnosno istraživača vrši se uz pomoć funkcije ATTR.

ATTR funkcija ima kao argumente koordinate y, x koji predstavljaju broj vrste i kolone pozicije ekrana (slično kao kod AT forme) čije se boje ispituju.

Broj koji je efekat funkcije ATTR je zbir sledeće četiri vrednosti:

— 128 ako je pozicija FLASH=1, 0 ako nije;
— 64 ako je pozicija BRIGHT=1, 0 ako nije;
— 8 * boja osnove ekrana (PAPER)
— INK boja

10.4. KORISNIČKA GRAFIKA

Govoreći o definisanju korisničkih grafičkih simbola napomenuli smo da je mnogo komfornije pri unosu programa konstante u DATA listi unositi u decimalnom nego u binarnom obliku. Da bismo došli do decimalnih brojeva bilo je potrebno ili vršiti računsku konverziju ili konsultovati odgovarajuću tabelu.

Sledeći program dozvoljava da se direktno dobiju decimalne vrednosti koje odgovaraju nekoj figuri nacrtanoj na ekranu. Rešetka je veličine 16 × 16, odnosno pomoću nje možemo odjednom opisati sadržaje 4 pozicije ekrana. Po pozicijama rešetke pomeraće se kursor korišćenjem dirki 5 — 8. Pri tome dirka 5 ima značenje pomeranja ulevo, 6 nadole, 7 gore i 8 udesno.

Pritiskanjem na dirku M definišemo sledeće funkcije: upis crnom bojom, upis belom bojom, pomeranje po rešetki bez promena boje, odnosno postojećeg stanja slike. Funkcija koja je važeća ispisuje se na dnu ekrana.

U momentu kada se figura nacrta na rešetki, pritiskom na dirku D aktiviramo programsko izračunavanje decimalnih vrednosti koje odgovaraju figuri i njihov prikaz na desnom delu ekrana. Takođe se na ekranu pojavi i definisani korisnički grafički simbol u pravoj veličini. Sve komande računar prima uz pomoć naredbe INKEY$.

Ovaj program je veoma koristan jer omogućava lak način definisanja grafičkih simbola, pri čemu se možete potpuno posvetiti izgledu simbola i ne trošiti vreme na zamorna i dosadna računanja.

Program

Grafika. U programu su definisani sledeći grafički simboli:

— Kursor se predstavlja grafičkim simbolom koji je dodeljen slovu "A".

```
 10 REM KORISNICKA GRAFIKA
 20 REM ----------------------
 30 DATA 1,125,125,125,125,125,
1,255,1,1,1,1,1,1,255
 40 DATA 255,255,255,255,255,25
5,255,255
 50 FOR I=0 TO 23
 60 READ A: POKE USR "A"+I,A
 70 NEXT I
 80 DIM M$(3,09): DIM B(18,18)
 90 LET M$(1)="POMERANJE": LET
M$(2)="CRNO": LET M$(3)="BELO"
 100 LET C=0: LET X=8: LET Y=9:
LET M=0
 190 REM ----------------------
 192 REM PRIKAZ GRAFICKOG
 194 REM GENERATORA - RESETKE
 196 REM ----------------------
 200 INK 0: PAPER 6: BORDER 6: C
LS
 210 PRINT: PRINT
 220 FOR I=1 TO 16
 230 PRINT " BBBBBBBBBBBBBBBB"
 240 NEXT I
 250 PLOT 7,32: DRAW 0,128: DRAW
128,0
 260 DRAW -64,0: DRAW 0,8
 270 PLOT 136,96: DRAW 8,0
 280 PRINT AT Y,X;"A"
 290 GO TO 1000
 380 REM ----------------------
 390 REM POMERANJE
 400 PAUSE 0: LET A$=INKEY$
 410 IF A$="M" THEN GO TO 1000
 420 IF A$="D" THEN PRINT AT Y,X
;CHR$ (145+B(X,Y)): GO TO 2000
 430 PRINT AT Y,X;CHR$ (145+B(X,
Y))
 440 IF A$="5" THEN LET X=X-1: I
F X=0 THEN LET X=16
 450 IF A$="8" THEN LET X=X+1: I
F X=17 THEN LET X=1
 460 IF A$="7" THEN LET Y=Y-1: I
F Y=1 THEN LET Y=17
 470 IF A$="6" THEN LET Y=Y+1: I
F Y=18 THEN LET Y=2
 480 IF M=2 THEN LET B(X,Y)=1: G
O TO 500
 490 IF M=3 THEN LET B(X,Y)=0
 500 PRINT AT Y,X;"A"
 510 GO TO 400
 980 REM ----------------------
 985 REM PROMENA REZIMA RADA
 990 REM ----------------------
1000 LET M=M+1: IF M=4 THEN LET
M=1
1010 PRINT AT 21,1;"BOJA-POMERAN
JE  ";M$(M)
1020 GO TO 400
1980 REM ----------------------
1985 REM KONVERZIJA BINARNO  U
1987 REM DECIMALNO
1989 REM----------------------
2000 FOR I=2 TO 17
2010 FOR J=1 TO 9 STEP 8
2020 LET D=0
2030 FOR K=0 TO 7
2040 IF B(K+J,I)=1 THEN LET D=D+
2↑(7-K)
2050 NEXT K
2060 PRINT AT I,23+J/2;D
2070 POKE USR "D" +I-2+(J-1)*2,D
```

```
2080 NEXT J: NEXT I
2090 PRINT AT 8,19;"DF";AT 9,19;
"EG"
2100 PRINT AT 12,18;"D F";AT 14,
18;"E G"
2110 GO TO 500
```

Sl. 10.4 — Program "Korisnička grafika"

— Bela pozicija u rešetki predstavlja se grafičkim simbolom koji je dodeljen slovu "B".

— Crna pozicija u rešetki predstavlja se grafičkim simbolom koji je dodeljen slovu "C".

Takođe se u programu, naredbama 2000—2080, slovima D—G dodeljuje sadržaj koji odgovara grafičkim simbolima formiranim na rešetki i datim sada u prirodnoj veličini. Primetite da za razliku od A—C koji dobijaju sadržaj preko naredbe DATA, ovde sadržaj zavisi od izgleda figure koju ste vi formirali.

Inicijalizacije. Naredbama 30 — 70 vrši se inicijalizacija pomenute grafike (A—C) i dodeljivanje početnih vrednosti promenljivim koje upravljaju igrom. To su:

— niz M$ čiji sadržaj postaju nazivi funkcija M$ (1) = POMERANJE, M$ (2) = CRNO, M$ (3) = BELO.

— X je horizontalna pozicija kursora. Ima početnu vrednost 8.

— Y je vertikalna pozicija kursora. Ima početnu vrednost 9.

— M$ je funkcija koja se trenutno obavlja (tekuća).

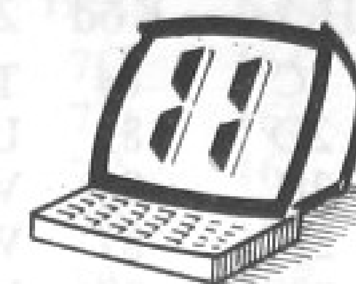
— B$ je niz koji sadrži podatke o boji svake ćelije na rešetki (0 — bela pozicija, 1 — crna pozicija).

Potom se naredbama 200 — 290 ispisuje slika rešetke.

Igra. Promena funkcije se vrši povećanjem vrednosti M za 1. Pri tome M može imati vrednosti 1, 2, i 3 (n 1000—1020).

Pomeranje kursora se obavlja naredbama 400—510. Ispitivanjem stanja tastature kursor se pomera u željenom pravcu, pri čemu se pozicija koja je ostala iza njega boji željenom bojom, zavisno od važeće funkcije. Promena boje se registruje u nizu B$.

U slučaju da se pritisne na dirku D, ide se na obradu skupa naredbi 2000 — 2110. Tu se, polazeći od sastava niza B, računaju odgovarajuće decimalne vrednosti koje odgovaraju grafičkim simbolima. Izračunate vrednosti se ispisuju na ekranu, a zatim se memorišu naredbom POKE USR, pri čemu se dodeljuju slovima D—G.



Prilozi

11.1. TABELA KODOVA I ZNAKOVA

Kôd	Znak	Heks	Kôd	Znak	Heks.
0	ne koristi se	00	22	AT upravljanje	16
1		01	23	TAB ,,	17
2		02	24	ne koriste se	18
3		03	25		19
4		04	26		1A
5		05	27		1B
6	PRINT COMMA	06	28		1C
7	EDIT	07	29		1D
8	KURSOR levo	08	30		1E
9	KURSOR desno	09	31		1F
10	KURSOR dole	0A	32	space (blanko)	20
11	KURSOR gore	0B	33	!	21
12	DELETE	0C	34	"	22
13	ENTER	0D	35	#	23
14	NUMBER	0E	36	$	24
15	NE KORISTI SE	0F	37	%	25
16	INK upravljanje	10	38	&	26
17	PAPER ,,	11	39	'	27
18	FLASH ,,	12	40	(	28
19	BRIGHT ,,	13	41	)	29
20	INVERSE ,,	14	42	*	2A
21	OVER ,,	15	43	+	2B

Kôd	Znak	Heks.	Kôd	Znak	Heks.
44	,	2C	84	T	54
45	—	2D	85	U	55
46	.	2E	86	V	56
47	/	2F	87	W	57
48	0	30	88	X	58
49	1	31	89	Y	59
50	2	32	90	Z	5A
51	3	33	91	[	5B
52	4	34	92	/	5C
53	5	35	93	]	5D
54	6	36	94	↑	5E
55	7	37	95	—	5F
56	8	38	96	£	60
57	9	39	97	a	61
58	:	3A	98	b	62
59	;	3B	99	c	63
60	<	3C	100	d	64
61	=	3D	101	e	65
62	>	3E	102	f	66
63	?	3F	103	g	67
64	@	40	104	h	68
65	A	41	105	i	69
66	B	42	106	j	6A
67	C	43	107	k	6B
68	D	44	108	l	6C
69	E	45	109	m	6D
70	F	46	110	n	6E
71	G	47	111	o	6F
72	H	48	112	p	70
73	I	49	113	q	71
74	J	4A	114	r	72
75	K	4B	115	s	73
76	L	4C	116	t	74
77	M	4D	117	u	75
78	N	4E	118	v	76
79	O	4F	119	w	77
80	P	50	120	x	78
81	Q	51	121	y	79
82	R	52	122	z	7A
83	S	53	123	{	7B

Kôd	Znak		Heks.	Kôd	Znak	Heks.
124	|		7C	164	(u)	A4
125	}		7D	165	RND	A5
126	©		7E	166	INKEY$	A6
127	□		7F	167	PI	A7
128	□		80	168	FN	A8
129	▝		81	169	POINT	A9
130	▘		82	170	SCREEN$	AA
131	▀		83	171	ATTR	AB
132	▗		84	172	AT	AC
133	▐		85	173	TAB	AD
134	▚		86	174	VAL$	AE
135	▜		87	175	CODE	AF
136	▖		88	176	VAL	B0
137	▞		89	177	LEN	B1
138	▌		8A	178	SIN	B2
139	▛		8B	179	COS	B3
140	▄		8C	180	TAN	B4
141	▟		8D	181	ASN	B5
142	▙		8E	182	ACS	B6
143	█		8F	183	ATN	B7
144	(a)		90	184	LN	B8
145	(b)		91	185	EXP	B9
146	(c)		92	186	INT	BA
147	(d)		93	187	SQR	BB
148	(e)		94	188	SGN	BC
149	(f)		95	189	ABS	BD
150	(g)		96	190	PEEK	BE
151	(h)		97	191	IN	BF
152	(i)		98	192	USR	C0
153	(j)		99	193	STR$	C1
154	(k)		9A	194	CHR$	C2
155	(l)	korisnikova	9B	195	NOT	C3
156	(m)	grafika	9C	196	BIN	C4
157	(n)		9D	197	OR	C5
158	(o)		9E	198	AND	C6
159	(p)		9F	199	<=	C7
160	(q)		A0	200	>=	C8
161	(r)		A1	201	<>	C9
162	(s)		A2	202	LINE	CA
163	(t)		A3	203	THEN	CB

Kôd	Znak	Heks.	Kôd	Znak	Heks.
204	TO	CC	230	NEW	E6
205	STEP	CD	231	BORDER	E7
206	DEF FN	CE	232	CONTINUE	E8
207	CAT	CF	233	DIM	E9
208	FORMAT	D0	234	REM	EA
209	MOVE	D1	235	FOR	EB
210	ERASE	D2	236	GO TO	EC
211	OPEN #	D3	237	GO SUB	ED
212	CLOSE #	D4	238	INPUT	EE
213	MERGE	D5	239	LOAD	EF
214	VERIFY	D6	240	LIST	F0
215	BEEP	D7	241	LET	F1
216	CIRCLE	D8	242	PAUSE	F2
217	INK	D9	243	NEXT	F3
218	PAPER	DA	244	POKE	F4
219	FLASH	DB	245	PRINT	F5
220	BRIGHT	DC	246	PLOT	F6
221	INVERSE	DD	247	RUN	F7
222	OVER	DE	248	SAVE	F8
223	OUT	DF	249	RANDOMIZE	F9
224	LPRINT	E0	250	IF	FA
225	LLIST	E1	251	CLS	FB
226	STOP	E2	252	DRAW	FC
227	READ	E3	253	CLEAR	FD
228	DATA	E4	254	RETURN	FE
229	RESTORE	E5	255	COPY	FF

11.2. MATRICE

Niz brojeva napisan u formi:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ a_{m11} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

zove se matricom reda $m \times n$. Zove se kvadratna matrica ako je $m=n$. Članove a_{ij} nazivamo elementima matrice. Ako je $n=1$, matrica se redukuje na vektor kolonu. Ako je $m=1$, matrica se redukuje na vektor vrstu. Kada je $n=m=1$, matrica postaje skalar. Matrica je kvadratna ako je $n=m$. Kvadratna matrica sa $a_{ii} = 1$, $i=1, \ldots, i$ m $a_{ij}=0, i \neq j$ označava se sa I i zove se jednačina matrica.

Definicije nekih osnovnih matričnih operacija:

Jednakost. Dve matrice A i B su jednake ako imaju isti broj vrsta i kolona i odgovarajuće jednake elemente, tj.:

$$a_{ij} = b_{ij}$$

Sabiranje. Zbir dve matrice koje imaju isti broj vrsta i kolona $(m \times n)$ je matrica $C(m \times n)$ kod koje je:

$$C_{ij} = a_{ij} + b_{ij}$$

$$A = \begin{bmatrix} 6 & 0 & 0 \\ 0 & 7 & 1 \\ 2 & 2 & 3 \end{bmatrix}, \quad B = \begin{bmatrix} 3 & 5 & 2 \\ 1 & 0 & 0 \\ 1 & 2 & 0 \end{bmatrix}, \quad C=A+B= \begin{bmatrix} 9 & 5 & 2 \\ 1 & 7 & 1 \\ 3 & 4 & 3 \end{bmatrix}$$

Množenje sa skalarom. Ako je α skalar, proizvod $\alpha \cdot A$ je matrica sa elementima αa_{ij}.

Množenje matrica. Ako je A $n \times r$ matrica, a B $r \times m$ matrica, proizvod AB je dat $n \times m$ matricom C gde je:

$$c_{ij} = \sum_{k=1}^{r} a_{ik} b_{kj}$$

Primer:

$$A = \begin{bmatrix} 6 & 0 & 0 \\ 0 & 7 & 1 \\ 2 & 2 & 3 \end{bmatrix}, \quad B = \begin{bmatrix} 3 & 5 & 2 \\ 1 & 0 & 0 \\ 1 & 2 & 0 \end{bmatrix}, \quad C=AB= \begin{bmatrix} 18 & 30 & 12 \\ 8 & 2 & 0 \\ 11 & 16 & 4 \end{bmatrix}$$

$$C_{31} = a_{31} \cdot b_{11} + a_{32} b_{21} + a_{33} \cdot b_{31}$$
$$= 2 \cdot 3 + 2 \cdot 1 + 3 \cdot 1 = 11$$

Množenje vektora i matrice. Ako je X 1 × n vektor (vektor vrste), a A n × m matrica, proizvod XA je 1 × m vektor Y, gde je:

$$y_i = \sum_{k=1}^{n} x_k \, a_{ki}$$

P r i m e r:

$$x = [\,1 \;\; 2 \;\; 3\,],\; A = \begin{bmatrix} 1 & 2 & 0 & 1 \\ 0 & 0 & 1 & 4 \\ 1 & 0 & 3 & 5 \end{bmatrix} \; Y = [\,4 \;\; 2 \;\; 11 \;\; 24]$$

$$y_4 = x_1 \cdot a_{14} + x_2 \cdot a_{24} + x_3 \cdot a_{34}$$
$$= 1 \cdot 1 + 2 \cdot 4 + 3 \cdot 5 = 24$$

P r i m e r r o t a c i j e:

$$[\,x' \; y' \; 1\,] = [\,x \; y \; 1\,] \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$x' = x \cdot x \cos\theta + y \cdot \sin\theta + 1 \cdot 0$$
$$y' = x \cdot (-\sin\theta) + y \cos\theta + 1 \cdot 0$$
$$1 = x \cdot 0 + y \cdot 0 + 1 \cdot 1$$

0	=	00000000	128	=	10000000
1	=	00000001	129	=	10000001
2	=	00000010	130	=	10000010
3	=	00000011	131	=	10000011
4	=	00000100	132	=	10000100
5	=	00000101	133	=	10000101
6	=	00000110	134	=	10000110
7	=	00000111	135	=	10000111
8	=	00001000	136	=	10001000
9	=	00001001	137	=	10001001
10	=	00001010	138	=	10001010
11	=	00001011	139	=	10001011
12	=	00001100	140	=	10001100
13	=	00001101	141	=	10001101
14	=	00001110	142	=	10001110
15	=	00001111	143	=	10001111
16	=	00010000	144	=	10010000
17	=	00010001	145	=	10010001
18	=	00010010	146	=	10010010
19	=	00010011	147	=	10010011
20	=	00010100	148	=	10010100
21	=	00010101	149	=	10010101
22	=	00010110	150	=	10010110
23	=	00010111	151	=	10010111
24	=	00011000	152	=	10011000
25	=	00011001	153	=	10011001
26	=	00011010	154	=	10011010
27	=	00011011	155	=	10011011
28	=	00011100	156	=	10011100
29	=	00011101	157	=	10011101
30	=	00011110	158	=	10011110
31	=	00011111	159	=	10011111
32	=	00100000	160	=	10100000
33	=	00100001	161	=	10100001
34	=	00100010	162	=	10100010
35	=	00100011	163	=	10100011

36 = 00100100	164 = 10100100
37 = 00100101	165 = 10100101
38 = 00100110	166 = 10100110
39 = 00100111	167 = 10100111
40 = 00101000	168 = 10101000
41 = 00101001	169 = 10101001
42 = 00101010	170 = 10101010
43 = 00101011	171 = 10101011
44 = 00101100	172 = 10101100
45 = 00101101	173 = 10101101
46 = 00101110	174 = 10101110
47 = 00101111	175 = 10101111
48 = 00110000	176 = 10110000
49 = 00110001	177 = 10110001
50 = 00110010	178 = 10110010
51 = 00110011	179 = 10110011
52 = 00110100	180 = 10110100
53 = 00110101	181 = 10110101
54 = 00110110	182 = 10110110
55 = 00110111	183 = 10110111
56 = 00111000	184 = 10111000
57 = 00111001	185 = 10111001
58 = 00111010	186 = 10111010
59 = 00111011	187 = 10111011
60 = 00111100	188 = 10111100
61 = 00111101	189 = 10111101
62 = 00111110	190 = 10111110
63 = 00111111	191 = 10111111
64 = 01000000	192 = 11000000
65 = 01000001	193 = 11000001
66 = 01000010	194 = 11000010
67 = 01000011	195 = 11000011
68 = 01000100	196 = 11000100
69 = 01000101	197 = 11000101
70 = 01000110	198 = 11000110
71 = 01000111	199 = 11000111

72 = 01001000	200 = 11001000
73 = 01001001	201 = 11001001
74 = 01001010	202 = 11001010
75 = 01001011	203 = 11001011
76 = 01001100	204 = 11001100
77 = 01001101	205 = 11001101
78 = 01001110	206 = 11001110
79 = 01001111	207 = 11001111
80 = 01010000	208 = 11010000
81 = 01010001	209 = 11010001
82 = 01010010	210 = 11010010
83 = 01010011	211 = 11010011
84 = 01010100	212 = 11010100
85 = 01010101	213 = 11010101
86 = 01010110	214 = 11010110
87 = 01010111	215 = 11010111
88 = 01011000	216 = 11011000
89 = 01011001	217 = 11011001
90 = 01011010	218 = 11011010
91 = 01011011	219 = 11011011
92 = 01011100	220 = 11011100
93 = 01011101	221 = 11011101
94 = 01011110	222 = 11011110
95 = 01011111	223 = 11011111
96 = 01100000	224 = 11100000
97 = 01100001	225 = 11100001
98 = 01100010	226 = 11100010
99 = 01100011	227 = 11100011
100 = 01100100	228 = 11100100
101 = 01100101	229 = 11100101
102 = 01100110	230 = 11100110
103 = 01100111	231 = 11100111
104 = 01101000	232 = 11101000
105 = 01101001	233 = 11101001
106 = 01101010	234 = 11101010
107 = 01101011	235 = 11101011

108	=	01101100	236	=	11101100
109	=	01101101	237	=	11101101
110	=	01101110	238	=	11101110
111	=	01101111	239	=	11101111
112	=	01110000	240	=	11110000
113	=	01110001	241	=	11110001
114	=	01110010	242	=	11110010
115	=	01110011	243	=	11110011
116	=	01110100	244	=	11110100
117	=	01110101	245	=	11110101
118	=	01110110	246	=	11110110
119	=	01110111	247	=	11110111
120	=	01111000	248	=	11111000
121	=	01111001	249	=	11111001
122	=	01111010	250	=	11111010
123	=	01111011	251	=	11111011
124	=	01111100	252	=	11111100
125	=	01111101	253	=	11111101
126	=	01111110	254	=	11111110
127	=	01111111	255	=	11111111

Sl. 11.1 — Decimalni i binarni brojevi

```
  10 REM DECIMALNI I BINARNI BRO
JEVI
  20 REM OD 0 DO 255
  30 FOR D=0 TO 127
  40 FOR Z=0 TO 1
  50 LET B=D+128*Z
  60 PRINT TAB (18*Z);B;TAB (4+1
8*Z);"= ";
  70 FOR X=0 TO 7
  80 LET O=INT(B/2↑(7-X))
  90 PRINT O;
 100 LET B=B-2↑(7-X)*O
 110 NEXT X
 120 NEXT Z
 130 PRINT
 140 NEXT D
```

Sl. 11.2 — Program "Decimalni i binarni brojevi"



Sadržaj

Mr Nenad Marković, dipl. inž. i Dušan Davidovac, inž. inf.: ZX Spectrum (Programiranje u BAZIC-u) — Recenzent: mr Aleksandar Zečević — Za izdavače: Branko Nikolić, direktor i glavni urednik i prof. mr Velimir Branković, direktor i glavni urednik — Urednici: Dušica Lučić i Svetislav Todić — Zajedničko izdanje: NIRO „Tehnička knjiga", Beograd, 7. juli 26 i Zavod za izdavanje udžbenika — OOUR Stvaranje i proizvodnja nastavnih sredstava, Beograd, Obilićev venac 5 — Štamparija: BIGZ, Beograd, Bulevar vojvode Mišića 17 — Tiraž: 5.000 primeraka — Oslobođeno poreza na promet na osnovu mišljenja Republičkog komiteta za kulturu SRS

Matematika: 4 osnovne računske radnje i matematičke funkcije

Časovnik: Ugrađen kvarcom kontrolisan digitalni časovnik sa štopericom, sa mogućnošću očitavanja stotih delova sekunde

Priključci:
- RF izlaz za priključak na standardni TV prijemnik (36. UHF kanal)
- Video-izlaz za priključak na monitor
- Izlaz za priključak kasetofona (kao spoljne memorije)
- Izlaz za EDGE konektor za priključak dodatne memorije i/ili perifernih jedinica, kao što su: štampač, generator zvuka, AD/DA konvertor i sl.

Pribor: Ispravljač, priključni kablovi, uputstvo za rukovanje

Dimenzije: 315×225×70 mm

Prodaja: Zavod za udžbenike i nastavna sredstva, Obilićev venac 5, 11000 Beograd, tel. 637-915 i 638-405

Servis: Elektronika inženjering, Karađorđev trg 11, 11080 Zemun, tel. 601-577

NAGRADNA IGRA

Dostavite ovaj kupon do 31. V 1985. na adresu NIRO »TEHNIČKA KNJIGA«, Beograd, 7. juli 26. U ponedeljak 10. VI 1985. u 12 časova u prostorijama NIRO »TEHNIČKA KNJIGA« obaviće se komisijski izvlačenje nagrada. Spisak nagrađenih biće objavljen u »TEHNIČKIM NOVINAMA« (broj 1 od 1. IX 1985.)

NAGRADE

- Mikroračunar »GALAKSIJA«
- 3 godišnje pretplate na časopis »MOJ MIKRO«
- 3 godišnje pretplate na časopis »GALAKSIJA«
- 3 godišnje pretplate na časopis »TEHNIČKE NOVINE«

KUPON (ZX SPECTRUM — PROGRAMIRANJE U BASIC-u

Ime i prezime ....

Ulica i broj ....

Broj pošte .... Mesto ....